Trabajo de fin de grado

Departamento de ingeniería Mecánica

Grado en Ingeniería en Tecnologías Industriales

Trabajo de Fin de Grado

Análisis numérico de las propiedades mecánicas del hueso cortical a tamaño micro

Agradecimientos

Gracias en primer lugar a mi tutor Miguel Marco Esteban, por guiarme a lo largo de todo este proyecto, no perder su interés en él en ningún momento y sobre todo por ser sincero en sus respuestas con cualquier asunto relacionado a su realización.

A toda persona con la que me haya encontrado en estos 4 años de periplo en la universidad, pues sin su contribución probablemente no estaría donde estoy. Especial agradecimiento a aquellos profesores que persiguieron como único objetivo el de enseñarnos a pensar, desarrollar nuestro ingenio y expandir nuestra mente más allá del logro de aprobar un examen.

Por último, pero no por ello menos importante, gracias a mi familia por su perseverancia, y a cualquier amigo que se haya interesado por mi y/o por el proyecto en algún momento del transcurso de su realización, sin vosotros tampoco sé si habría podido sobrellevarlo.

Resumen

En este trabajo se ha logrado crear desde cero un método que genere un volumen representativo de tejido cortical, sobre el cual se imponen numerosos estados de carga, con el objetivo de realizar un análisis de cada uno de ellos. Esto nos permitirá obtener los datos necesarios para poder calcular el valor de todas las constantes elásticas propias de un material anisótropo, y comprender así el comportamiento elástico de este tipo de tejido.

Para la creación del modelo se han utilizado principalmente dos herramientas: el programa de *Computer Aided Engineering: Abaqus CAE,* que utiliza el metodo de los elementos finitos para resolver problemas de ingeniería mecánica y estructural, y Python, un lenguaje de programación multiusos que mediante scripts nos ha permitido modificar a nuestro antojo le ejecución de ciertas funciones de Abaqus, para así automatizar la generación del modelo todo lo posible.

Se ha utilizado un total de 3 scripts para crear el modelo: el primero está centrado en generar automáticamente la distribución de las osteonas (unidad básica estructural del tejido cortical), de posición aleatoria, y sin contacto entre ellas, creadas una a una hasta que estas ocupen el 65 por ciento del volumen del modelo. El segundo script se encargaba principalmente de asignar las secciones de los materiales a las distintas partes del modelo para facilitar esta tarea y así ahorrar una importante cantidad de tiempo al usuario. Los distintos estados de carga se introducción manualmente para poder modificarlos en cualquier momento, al igual que se seleccionaban manualmente los parámetros que se desea analizar y obtener sus resultados correspondientes. Por último el tercer script se encargaba de realizar el mallado del modelo, crear el análisis y correrlo.

Se ha llevado a cabo un análisis de la sensibilidad de la malla y de la influencia del tamaño del RVE, para crear un modelo que combine precisión y un mínimo coste computacional posible.

Una vez conocidas todas las dimensiones del modelo se han creado 5 modelos finales sobre los que realizar los análisis pertinentes: un total de 45 análisis

distintos a contrastar, para obtener los modulos de Young, coeficientes de Poisson y módulos de cortadura de los 5 modelos en todas sus direcciones.

Hemos logrado de esta forma obtener todas las constantes que nos permiten valorar la respuesta mecánica en régimen elástico del tejido cortical.

Palabras clave: osteona, canal de Havers, método de los elementos finitos, Abaqus, Python, script, módulo de Young, coeficiente de Poisson y módulo de cortadura.

Abstract

On this Project we have managed to create from the start a method that generates a representative volume of cortical bone, on which certain loading states are imposed, with the goal of performing an analysis for each one of them. This will allow us to obtain the data necessary to calculate the value of all the elastic constants related to an anisotropic material, so then we get a better understanding of the elastic behavior of this type of tissue.

For the creation of the model two tools have been used mainly: the *Computer Aided Engineering:* Abaqus CAE, which uses the finite element method to solve mechanical and structural engineering related problems, and Python, a multi-functional programming language which by using scripts has allowed us to modify the execution of certain Abaqus functions just as we please, so that we can automate the creation of the model as much as we could.

A total of 3 scripts have been used to create the model: the first one is focused on generating automatically the distribution for the osteons (basic level structure of cortical bone tissue), arranged on random positions, without contact between them, and created one by one until the 65& of the volume fraction of the model is occupied by osteons. The second script focuses on relating each of the material sections with is correspondent part of the model in order to make this an easier task and save up quite some time to the user. The different loading states are introduced manually so they can be modified whenever it is required, just as also the parameters meant to be analyzed were selected manually to obtain the correspondent results. Finally the third script concentrates on generating and implementing the mesh for the model, create the analysis and run it.

A mesh sensitivity analysis has been performed and also one that determines the influence of the size of the RVE, so that the model created manages to be a combination of precision on the results and the lowest computer expense possible.

Once all the dimensions of the model are known, 5 final models have been created, in order to perform all the analysis required: a total of 45 different analysis have been done, so making a contrast with them is possible, to obtain the Young´s modulus, the Poisson´s coefficients and the shear modulus of the 5 different models in all the directions needed.

We, this way, have achieved to obtain all the constants that will let us evaluate the mechanical response on the elastic regime of the cortical bone tissue.

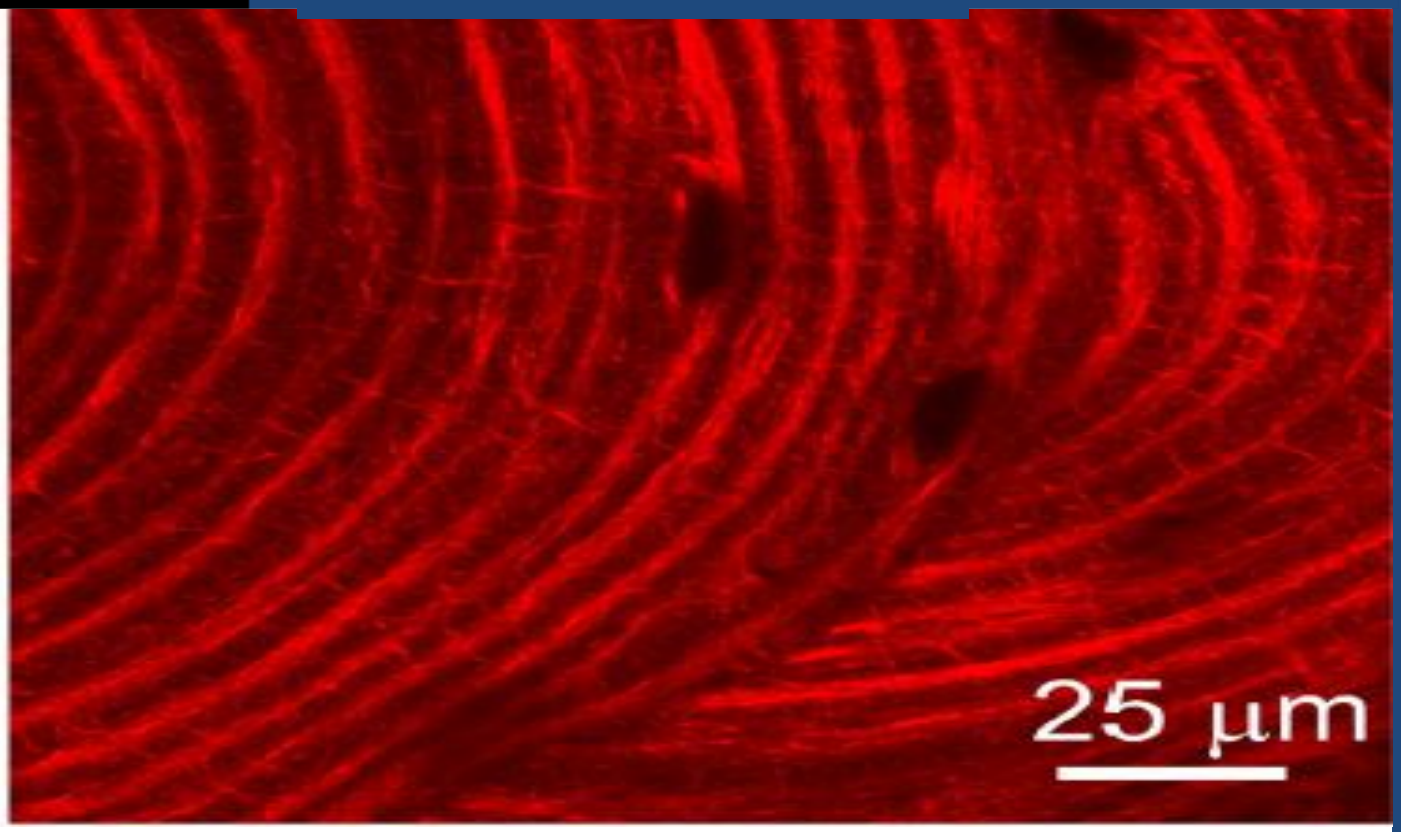
Keywords: osteon, Havers canal, finite elements method, Abaqus, Python, script, Young´s modulus, Poisson´s coefficient and shear modulus.

Índice:

Índice de figuras

Índice de tablas:

Capitulo I: Introducción



1.1 Motivación

En el mundo presente, gracias a las nuevas tecnologías, que se hallan en constante evolución, se han conseguido desarrollar ramas del conocimiento y la ingeniería como la biomecánica, disciplina dedicada al estudio de las consecuencias mecánicas que suponen para el ser humano las acciones que ejecuta al moverse, hacer deporte o incluso simplemente sestando de pie. Es por tanto la biomecánica una combinación de diversas ramas como la anatomía y la biología con la ingeniería mecánica y estructural [19].

La biomecánica plantea buscar soluciones tecnológicas a problemas derivados de las solicitaciones mecánicas, como por ejemplo, crear materiales tecnológicos con las propiedades apropiadas que puedan ser utilizados en prótesis que sustituyan huesos, crear dispositivos de apoyo para procesos de rehabilitación en caso de pérdida de la capacidad motriz, etc.

Concretamente en este trabajo, gracias al desarrollo de sofisticados programas que facilitan el cálculo estructural y permiten la resolución de complejos problemas que utilicen el método de elementos finitos para esa resolución, hemos planteado la opción de obtener mediante el uso de estos programas las propiedades mecánicas de un volumen representativo de tejido óseo cortical. Este planteamiento cobra sentido, ya que este tejido se halla en porciones de muy pequeño tamaño en el cuerpo humano, por lo que realización de un ensayo mecánico se torna como una solución difícil para llevar a cabo y estudiar el comportamiento mecánico de este tejido.

Si este trabajo se realiza satisfactoriamente, son numerosas las puertas que abre a futuros trabajos relacionados con el presente proyecto.

1.2 Objetivos

El principal objetivo de este trabajo es la creación de un modelo que represente fielmente un volumen representativo de una porción de tejido óseo cortical, el cual supone un 80 por ciento aproximadamente del tejido óseo presente en los

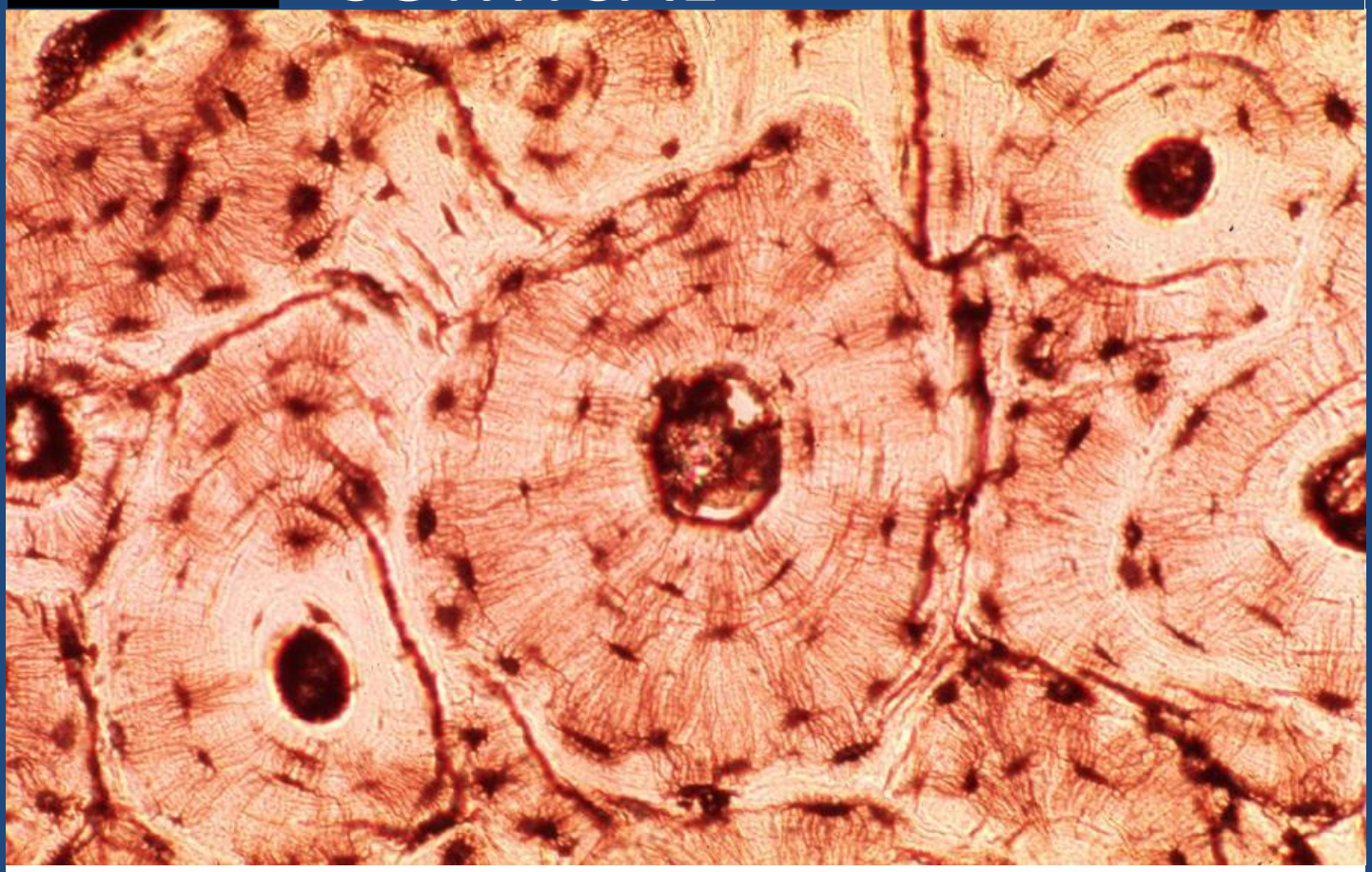
huesos del esqueleto humano [2]. Es por ello necesario que el modelo nos permita obtener los valores de todas las constantes elásticas requeridas para un material anisótropo, con distintos valores en direcciones Z, X e Y.

Los resultados han de ser precisos, por lo que será imprescindible, como en cualquier problema que se base en el método de los elementos finitos realizar un análisis de la sensibilidad de la malla con elementos de distinto tamaño y analizar de forma análoga como influye e tamaño del volumen representativo en los resultados obtenidos.

El programa utilizado que nos servirá de base será Abaqus, dando lugar a otro objetivo: apoyándonos en el lenguaje de programación Python, conseguir mediante la implementación de scripts crear una forma rápida y fácil de generar un modelo como el deseado, con una distribución de osteonas aleatoria, pero con ciertas condiciones que logren que el modelo represente de manera fidedigna la realidad.

Los resultados además han de ser consistentes, por lo que no valdrá realizar un solo modelo, sino que serán 5 los creados en total para obtener una mejor visión del comportamiento del hueso cortical, cumpliendo con el objetivo de permitir obtener de este trabajo un modelo de material que unifique los 3 componentes que constituyen el hueso cortical y determinan su comportamiento mecánico: la matriz intersticial, las osteonas y las líneas cementantes.

CAPITULO II: EL HUESO CORTICAL



En este capítulo se expone la información que es imprescindible conocer sobre los huesos del cuerpo humano y el tejido óseo, y sus propiedades mecánicas. También se exponen las distintas diferencias que existen entre los dos tipos de hueso del cuerpo humano: cortical y trabecular, haciendo por último especial hincapié en la definición del hueso cortical, sobre el que gira este proyecto

2.1 El esqueleto Humano

El esqueleto humano compuesto de alrededor de unos 206 huesos en una persona adulta, realiza una serie de funciones mecánicas y biológicas indispensables para la vida humana, siendo las de mayor relevancia [10]:

-**Soporte:** El esqueleto constituye la estructura de sustento del cuerpo humano, teniendo que ser capaz el peso de órganos y músculos, además de los esfuerzos que la actividad de estos últimos pueda requerir.

-**Protección:** El esqueleto confiere protección frente a daños físicos a los distintos órganos del cuerpo humano. Distintos grupos de huesos protegen distintos grupos de órganos, por ejemplo, la columna vertebral protege la médula espinal, el cráneo la parte central del sistema nervioso, etc.

-**Movimiento:** Los huesos en conjunto con los músculos, permiten nuestro movimiento al contraerse estos últimos.

-Depósito de sales minerales: Aproximadamente el 99% del calcio y el 80% del fósforo de nuestro cuerpo se encuentra en los huesos. El calcio participa en distintas funciones inmunitarias, nerviosas y musculares, mientras que el fósforo es un constituyente del material genético y participa en el metabolismo energético.

-Hematopoyesis: Es en el interior de ciertos huesos (aquellos que contienen la medula espinal) donde tiene lugar este proceso que da lugar a la formación de nuevas células sanguíneas.

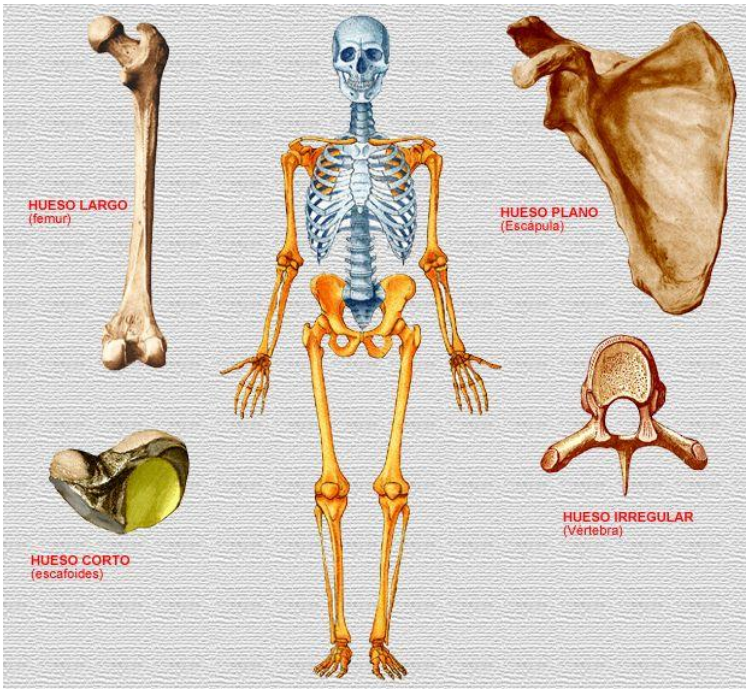


Figura 1: Imagen del esqueleto humano y clasificación de los distintos tipos de huesos según su fisonomía [12].

2.1.1 Composición del tejido óseo

El esqueleto está formado en su mayoría por el tejido óseo. Este tejido es un tejido dinámico en el sentido de que se halla constantemente en un proceso de renovación constante, mediante la resorción ósea (desintegración del tejido óseo) y la consecuente remodelación ósea.

Es además un tejido conjuntivo, compuesto en gran medida por una matriz compuesta de agua, proteínas y sobre todo sales minerales. Además de esta matriz, en el hueso hay presente en alrededor de un 2% de su composición distintos tipos de células [2,12]:

-**Osteoblastos** (células destructoras de hueso): Son células polinucleadas que derivan de la fusión de elementos monocíticos de la medula ósea, localizados en la superficie del hueso. Participan en el proceso de homeostasis, ayudando a regular los niveles de calcio, pero su principal función es la de eliminar tejido óseo que ha perdido sus propiedades resistentes y elásticas para iniciar así el

proceso de renovación ósea. Esta destrucción se produce mediante la liberación de enzimas como las fosfatasas ácidas, que acidifican la superficie del hueso, provocando la resorción ósea [2,12,32].

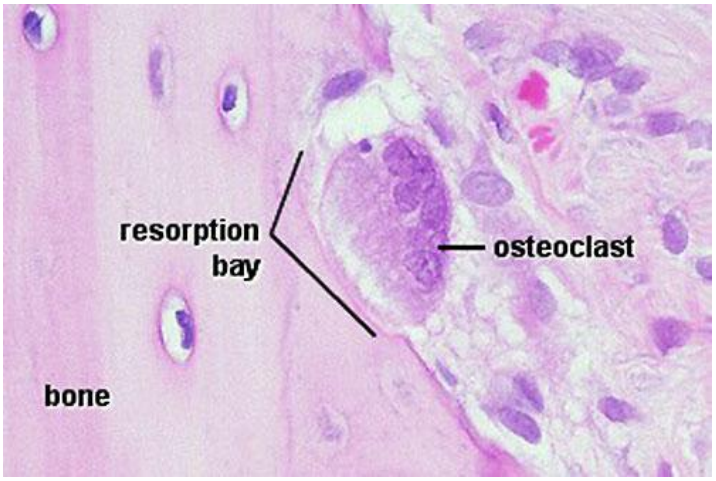


Figura 2: Imagen de un osteoclasto, que da lugar a una "bahía de resorción" en su proceso de destrucción del tejido óseo [30].

-Osteoblastos (células formadoras de hueso): Son células mononucleares que provienen de células madre, que dan lugar a pre-osteoblastos que o bien se convierten en osteoblastos maduros, o bien en células de revestimiento óseo. En caso de no convertirse en ninguno de los anteriores tipos, se destruyen por apoptosis (muerte celular provocada por estímulos intra o extracelulares.

Los osteoblastos se sitúan sobre la superficie ósea y allí llevan a cabo su principal función de regeneración del hueso. Este proceso tiene lugar cuando la célula recibe el estímulo adecuado y comienza a generar colágeno principalmente, pero también osteonectina y fosfatasa alcalina dando lugar al tejido osteoide. Cuando el nuevo tejido óseo formado va desarrollándose y expandiéndose, algunos osteoblastos quedan envueltos en él, conviritendose en osteocitos que se dedicarán al mantenimiento del hueso. Los osteoblastos también colaboran en la regulación de la mineralización de la matriz [2,12,32].

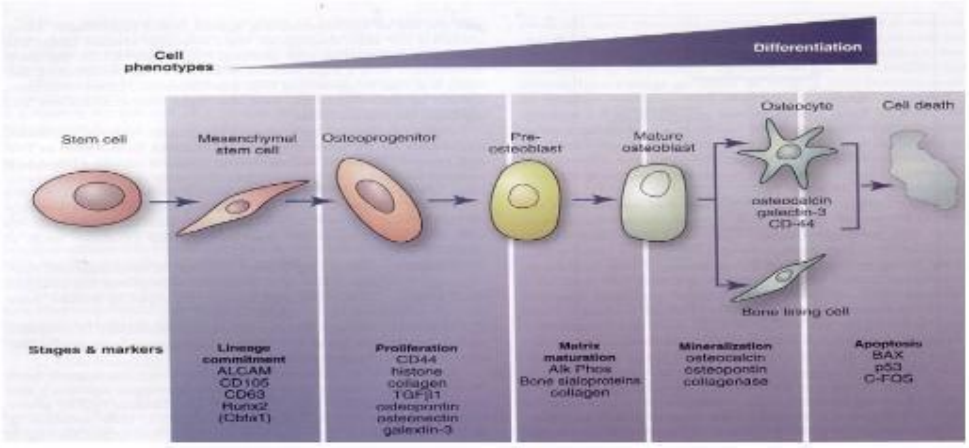


Figura 3: Imagen que representa el ciclo de formación de un osteoblasto y osteocito [2].

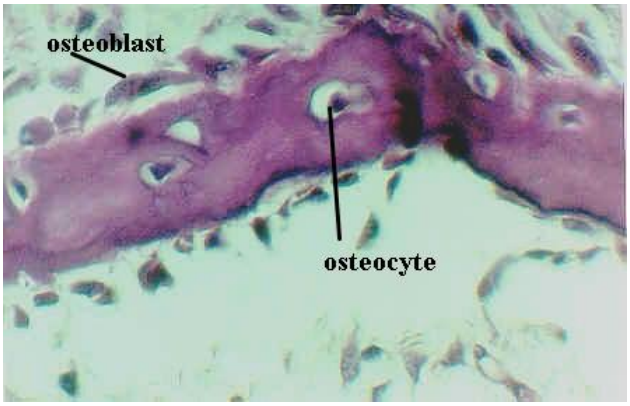


Figura 4: Imagen que muestra la localización de osteoblastos y de osteocitos (osteoblastos atrapados en el tejido óseo formado) [16].

-Osteocitos (células maduras del hueso): Son células derivadas de los osteoblastos formadas tras la mineralización del osteoide. Aunque solo un 15-30%de los osteoblastos acaba convirtiéndose en un osteocito, estos constituyen el 90% de las células de un hueso adulto. Los osteocitos se encuentran en las denominadas lagunas osteocitarias, y su función principal es la homeostasis mineral. También se encarga de llevar a cabo actividades de las células del tejido óseo como el intercambio de nutrientes y desechos.

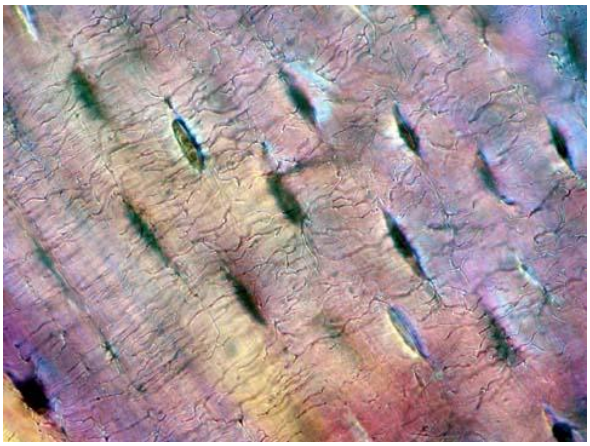


Figura 5: Imagen en que se aprecian distintos osteocitos (manchas oscuras en la imagen) distribuidos en la matriz ósea [16].

La matriz ósea por su parte se compone de:

-Fracción orgánica: Supone el 30% de la matriz ósea y está compuesta en gran parte (alrededor del 90%) de colágeno, rico en aminoácidos como la hidroxiprolina. Este colágeno se dispone en forma de triple hélice (3 fibras entrelazadas), dando lugar a microfibrillas, que forman haces, que dan lugar a su vez a fibras colágenas. Estas fibras colágenas son las que confieren a la matriz su resistencia a los esfuerzos de tracción.

El 5-10% restante lo conforman proteínas no colágenas generadas por los osteoblastos. Las más importantes son la osteocalcina, la osteonectina y la osteopontina, que actúan como marcadores bioquímicos de la homeostasis del calcio y de la actividad ósea. También se hallan presentes proteínas de crecimiento tales como las proteínas óseas morfogenéticas[2,12,32].

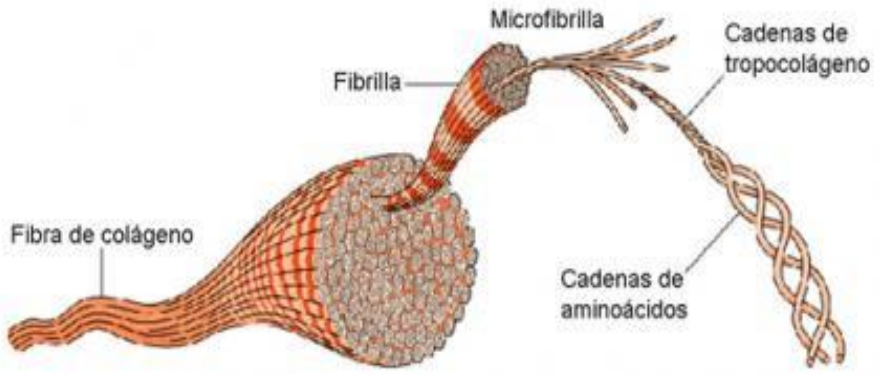


Figura 6: Imagen que muestra la estructura de una fibra de colágeno y todos los niveles estructurales en que se subdivide [31].

-Fracción inorgánica: la fracción mineral supone el otro 70% de la matriz ósea. Está formada mayormente por hidroxiapatita en forma de cristales formados por un 80% de fosfato tricálcico, un 10% de carbonato cálcico y el resto son impurezas e inclusiones que modifican las propiedades mecánicas de la matriz. Los cristales aportan unas excelentes cualidades mecánicas y se encargan principalmente de soportar los esfuerzos a compresión. Estos cristales se encuentran situados entre las fibras colágenas dando lugar a una estructura bifásica en que se reparten los distintos esfuerzos para optimizar la respuesta mecánica del tejido óseo [2,12].

2.1.2 Propiedades mecánicas

Se puede afirmar que el tejido óseo es heterogéneo y anisótropo puesto que sus propiedades mecánicas dependen de diversos factores como la edad y el género, la localización de cada hueso,etc.

Debido a los procesos de remodelación ósea constantes, los huesos se adaptan localmente, de forma que resisten mejor las cargas a las que suelen estar sometidos, por ejemplo los huesos de la pierna presentan unas mejores propiedades mecánicas a compresión mientras que los del brazo por su parte se comportan mejor frente a esfuerzos de tracción.

Las propiedades del hueso también se ven afectadas por el grado de porosidad y mineralización. Los huesos pueden llegar a alcanzar un nivel de porosidad del 90%, y aunque el tejido óseo mantenga sus propiedades, una mayor porosidad supone una disminución considerable de sus propiedades mecánicas, ya que para una muestra del mismo volumen, hay una menor proporción de tejido óseo [9].

El grado de mineralización también influye notablemente en las propiedades del hueso. Al aumentar el nivel de mineralización, el hueso se vuelve más resistente mecánicamente, pero en cambio también aumenta su fragilidad y disminuye su ductilidad. Un hueso más mineralizado por tanto es mucho más vulnerable a impactos, pudiendo fracturarse más fácilmente. El grado de mineralización no suele ser un problema, ya que suele mantenerse entre determinados valores, por lo que es la porosidad lo que a efectos prácticos tiene una mayor influencia en las propiedades óseas [32].

La anisotropía del hueso se refiere a que las propiedades varían dependiendo de la dirección en que sean analizadas. Esta anisotropía es diferente según el tipo de tejido óseo, como se explica en punto que viene a continuación

2.2 Hueso cortical y trabecular

El tejido óseo se clasifica en dos tipos según su porosidad y la unidad básica estructural de que se componen: el hueso cortical y el hueso trabecular.

El hueso cortical es mucho más denso, con una porosidad de entre tan solo un 5 y un 10%. Este tejido supone aproximadamente el 80% del tejido óseo de un esqueleto adulto y se localiza principalmente en la parte externa de los huesos así como en gran parte de la diáfisis de los huesos largos. El hueso cortical es el destinado a soportar los mayores esfuerzos mecánicos y es menos activo metabólicamente en comparación con el trabecular.

El hueso trabecular por otro lado puede alcanzar un grado de porosidad de alrededor del 50 por ciento llegando incluso a valores del 90%. Este tipo de

hueso está situado principalmente en el interior de huesos planos, en las vértebras y en los extremos de los huesos largos (epífisis). Tiene una superficie unas 20 veces superior a la del hueso cortical y metabólicamente tiene una actividad unas 8 veces superior (su tasa de renovación anual ronda el 25%, un 2-3% más que el hueso cortical). Tal actividad metabólica lo vuelve más susceptible a variaciones nutricionales, hormonales y bioquímicas, favoreciendo la aparición de fenómenos como la osteoporosis [9, 32].

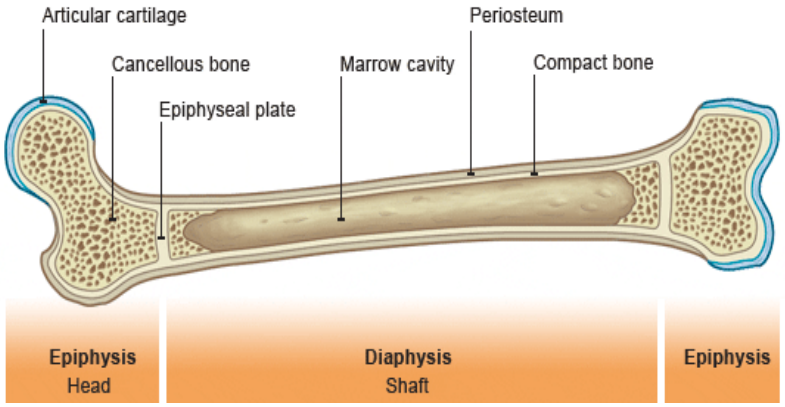


Figura 7: Imagen que muestra de forma esquemática las distintas localizaciones de hueso cortical y trabecular en un hueso largo [33].

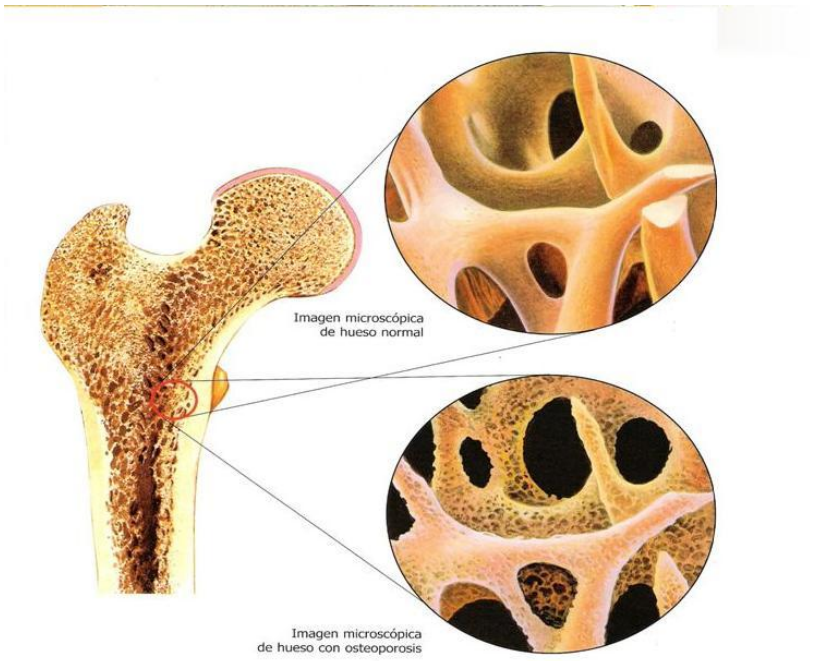


Figura 8: Imagen en que se observa una comparativa visual entre un hueso sano (arriba) y uno que sufre osteoporosis (abajo), volviéndose de esta forma más propenso a la fractura [34].

2.2.1 Arquitectura y composición del hueso trabecular

El hueso trabecular(o hueso esponjoso) está constituido por una red de pequeñas placas y varillas irregulares que se unen y entrelazan entre sí, conformando su unidad básica estructural, las trabéculas óseas. Las trabéculas están formadas por laminillas óseas las cuales contienen numerosos osteocitos. Entre los espacios vacíos de las trabéculas, se puede encontrar tejido graso y medula ósea roja, que como se ha dicho anteriormente se encarga de originar nuevas células sanguíneas mediante hematopoyesis.

Las distintas trabéculas se orientan de una forma u otra según el esfuerzo al que estén sometidas, ofreciendo resistencia a compresión y cortadura

principalmente. Debido a las diferentes orientaciones, en determinadas partes de huesos se forma lo que se conoce como triangulo de Ward, un área de menor densidad ósea que pese a ser anatómicamente normal, supone una fragilidad extra[12,2,32].

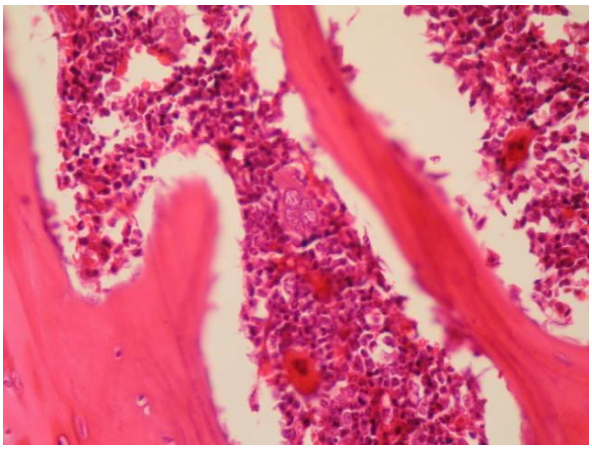


Figura 9: Imagen del tejido cortical, con numerosas trabéculas óseas y tejido hematopoyético [16].

Figura 10: Imagen que ilustra el triángulo de Ward en el trocánter del fémur [2].

2.3 Arquitectura y funcionalidad del hueso cortical

Como este trabajo se centra en el hueso cortical profundizaremos un poco más en la descripción de las distintas funciones que cumple y las diferentes partes que conforman su anatomía:

2.3.1 Funciones

Las principales funciones del hueso cortical son las siguientes[10]:

-**Soporte:** el tejido cortical por ser considerablemente más denso que el esponjoso, es el principal encargado de soportar los esfuerzos mecánicos a que se exponen los huesos

-Rigidez: Como este tipo de hueso se localiza normalmente en la zona exterior de los huesos, es el encargado de dotar a los huesos de resistencia y rigidez.

-Contenedor del suministro de sangre: Por el interior del hueso cortical circulan vasos sanguíneos que se encargan del suministro de sangre y facilitan el intercambio de nutrientes con las células presentes en el hueso.

-**Transporte de desechos:** Por el hueso cortical también discurren los vasos linfáticos que transportan la linfa, un líquido que transporta desechos y distintas células del sistema inmunitario.

2.3.2 Sistema de Havers

Los sistemas de Havers (también conocidos como osteonas) son la unidad estructural básica del hueso cortical, y está formada principalmente por un conducto central (canal de Havers), rodeado por varias laminas concéntricas entre sí y concéntricas también con el canal de Havers.

Por el canal de Havers discurren vasos sanguíneos, linfáticos y nervios, encargándose estos últimos de irrigar y transmitir los impulsos pertinentes a las células óseas (osteocitos y osteoblastos). Esto es posible gracias unos canalículos que nacen en el canal de Havers y se sitúan de forma radial, en dirección a las lagunas de donde se localizan los osteocitos. Los osteocitos introducen sus prolongaciones para interactuar con el canal de Havers y otras células y así poder absorber los nutrientes necesarios.

Las láminas situadas en torno al conducto de Havers se orientan en función de las solicitaciones mecánicas que soportan, siendo capaces de variar esta orientación si varían los esfuerzos. Esto es posible gracias al proceso de renovación continua de los huesos (resorción ósea) anteriormente detallado. Las láminas están formadas de tejido óseo mineralizado y suele haber en torno a entre 4 y 20 de ellas. Cada lámina tiene sus fibras de colágeno situadas paralelamente y es la dirección de estas fibras la que determina la resistencia mecánica de la matriz ósea.

Las osteonas están comunicadas entre sí gracias a unos canalículos dispuestos de forma transversal conocidos como conductos de Volkmann. Además cada osteona está delimitada por una capa calcificada conocida como línea cementante.

Por último encontramos el periostio que recubre exteriormente a todo el conjunto de osteonas. El periostio es la parte más externa de un hueso, y desde su parte interna nacen los vasos sanguíneos y se producen osteoblastos que generarán el tejido óseo renovado. El periostio se une con el tejido cortical mediante unas prolongaciones conocidas como fibras de Sharpey que proporcionan una cohesión suficientemente buena como para afianzar esta unión [15].

Figura 11: Imagen que representa las distintas orientaciones adoptadas por las láminas de forma que adquieren una mayor resistencia mecánica [2]

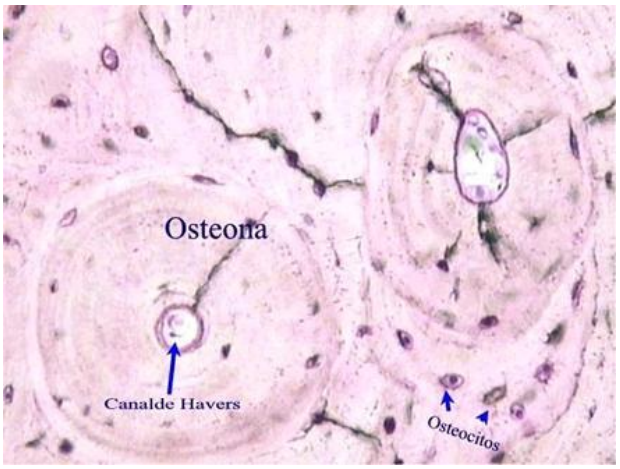


Figura 12: Imagen microscópica de una osteona real en la que se pueden apreciar sus elementos más representativos: el canal de Havers, las láminas que lo rodean y la línea cementante [15].

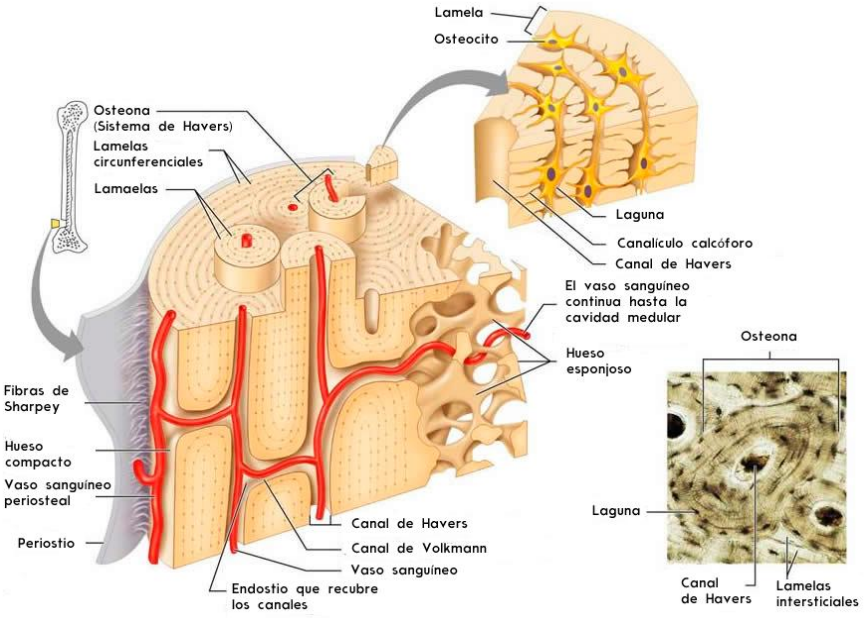
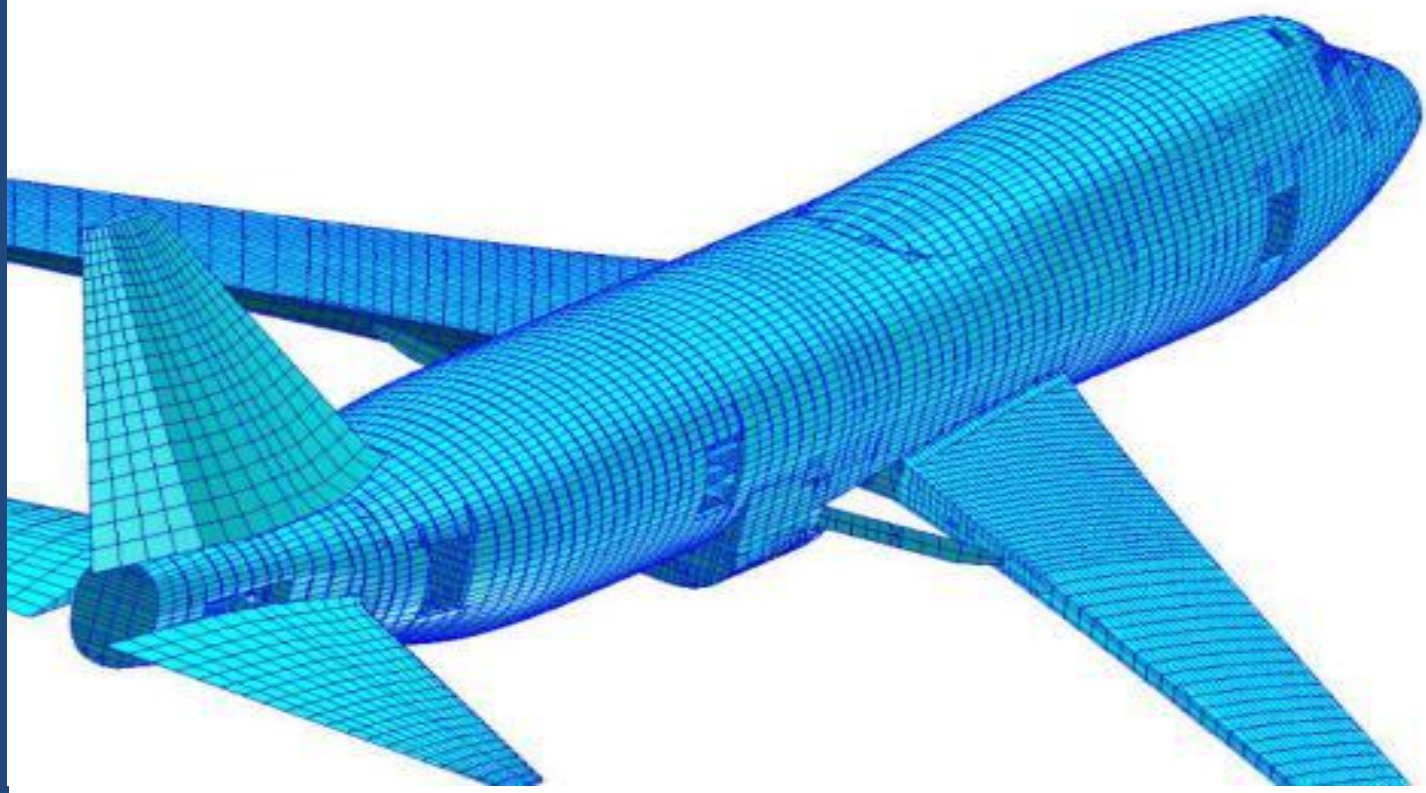


Figura 13: Imagen que ilustra el esquema de una sección de hueso en que se puede ver la disposición todos los elementos de un sistema de Havers, como los canales de Volkmann, y también las uniones entre tejido cortical, trabecular y el periostio (mediante las fibras de Sharpey) [35].

CAPITULO III: DISEÑO DEL MODELO



En este capítulo se expondrá todo lo relacionado con la definición del modelo que nos permitirá simular una muestra representativa del hueso cortical, además de la información sobre las herramientas y programas que nos servirán para crear dicho modelo y obtener los resultados deseados.

3.1 Definición del modelo

Para conseguir representar fielmente el tejido cortical de forma que los resultados del proyecto sirvan de guía para facilitar la comprensión de su comportamiento mecánico se ha elegido crear un modelo numérico que se base en el método de elementos finitos ya que las dimensiones de incluso las mayores superficies de este tejido halladas en el cuerpo humano son del orden de apenas varios milímetros, lo que dificulta la realización de ensayos mecánicos para obtener sus propiedades mecánicas.

La idea básica del modelo es crear un cubo que haga las veces de volumen representativo de una muestra de tejido cortical. Se supondrá que el cubo creado en un principio representa la matriz ósea intersticial y sobre él se harán una serie de modificaciones que generen la estructura básica de los sistemas de Havers. Por simplicidad se ha decidido no incluir los canales de Volkmann y se supone que es una zona de tejido cortical aislado, es decir, no se incluye ni tejido trabecular adyacente ni el periostio, para obtener de este modo las propiedades del tejido cortical exclusivamente.

De este modo finalmente quedan incluidas una serie de diferentes osteonas distribuidas a lo largo del cubo incluyendo cada una un conducto vacío central (canal de Havers), una zona concéntrica alrededor que simule las laminillas y alrededor una última capa concéntrica en representación de la línea cementante.

Como dice la teoría las laminillas de cada osteona pueden variar en número entre 4 y 20 aproximadamente, en el modelo se crearan distintas osteonas con distintos tamaños, dentro de unos determinados límites dimensionales. Esta condición también se aplicará al canal de Havers y la línea cementante, ya que

cada osteona tiene distinto tamaño, al interactuar unas con otras y el tejido que las rodea.

Otra condición que se impondrá será que no exista contacto o solape entre distintas osteonas, ya que esto no se produce en la realidad, permitiéndoles evidentemente situarse a distancias muy pequeñas.

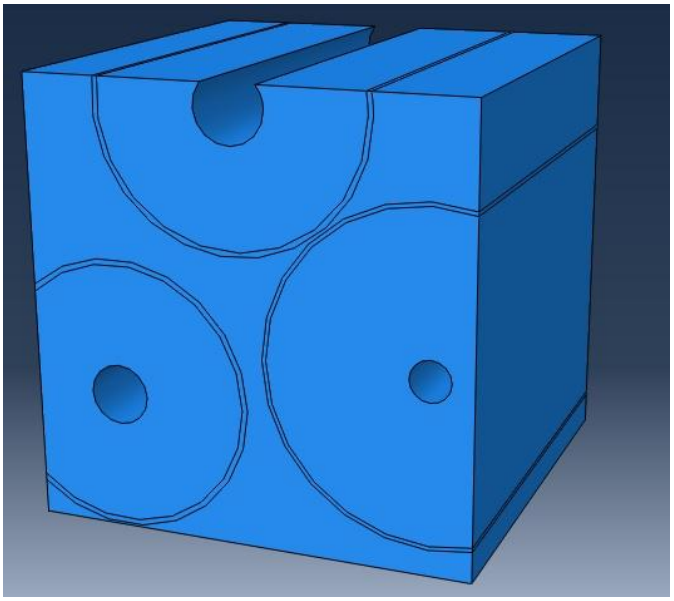


Figura 14: Imagen con la representación básica del modelo que se desea obtener [obtención propia]

3.2.1 Dimensiones

Las dimensiones de tanto el canal de Havers como de las osteonas y la línea cementante se han obtenido mediante investigación bibliográfica. Tras contrastar varios artículos se decidió que el tamaño de canales de Havers que se usarían para el modelo sería de un tamaño medio de 35µm, con un valor máximo de 60 µm. Las osteonas tendrán un tamaño medio de unos 140 µm, con un valor máximo de 250 µm. Por último las líneas cementantes tendrán un valor medio de 5 µm con un máximo de 10 µm. Todos estos valores fueron obtenidos experimentalmente mediante observación microscópica.

También se ha usado la bibliografía para conocer la fracción de volumen correspondiente a matriz intersticial y el conjunto de osteonas. En concreto para nuestros modelos se usara un valor de una fracción de volumen correspondiente a las osteonas del 65% [4].

Para esta fracción de volumen y este tamaño de los canales de Havers, la porosidad debería rondar el 5-10 por ciento, acorde con la descripción del tejido cortical del capítulo II.

El tamaño del lado del cubo que constituye el volumen representativo será obtenido más adelante, al realizar el análisis numérico de la variación de los resultados en función del tamaño del cubo [4].

Tabla resumen de las dimensiones del modelo [4]

Componente	Tamaño medio(µm)	Tamaño máximo(µm)
Osteonas:	140	250
Canal de Havers:	35	60
Linea cementante:	5	10

3.2.2 Propiedades

Es sabido de antemano que macroscópicamente el modelo representa un material transversalmente ortótropo, siendo la dirección preferente la paralela a la dirección de las osteonas. No obstante el análisis se realiza de tal forma que se obtengan todas las constantes elásticas en todas sus direcciones, como si de un material ortótropo se tratara, esperando evidentemente valores similares para constantes que se pueden asumir equivalentes (por ejemplo el coeficiente de Poisson en dirección xz y zx).

Por simplicidad se asumirá que trabajamos con un material homogéneo y se definirá un comportamiento elástico isótropo para cada uno de los componentes de la matriz.

Para determinar las propiedades que deben ser asignadas a cada componente del modelo también se ha realizado una investigación bibliográfica. En el artículo consultado las propiedades se obtuvieron realizando una serie de medidas mediante nano-indentación en probetas de hueso de un tamaño del orden de varios milímetros, obteniendo el módulo de Young local tanto en dirección longitudinal como transversal para más de cien osteonas. La nano-indentación es un tipo de ensayo pensado para muestras de pequeño tamaño, que consiste básicamente en presionar con una fuerza conocida una punta de material, generalmente de gran dureza y cuyas propiedades son también conocidas, contra la muestra cuyas propiedades se desean obtener. Con este ensayo se obtiene una curva carga-desplazamiento, a partir de la cual se puede calcular el módulo de Young mediante la siguiente formula:

$$E_r = \frac{1}{\beta}\frac{\sqrt{\pi}}{2}\frac{S}{\sqrt{A_p(h_c)}}$$

Donde A_p es el área proyectada a la profundidad h_c, S es la dureza del contacto teniendo en cuenta la dureza del material de la punta y la respuesta del dispositivo de medida y β es una constante geométrica[18].

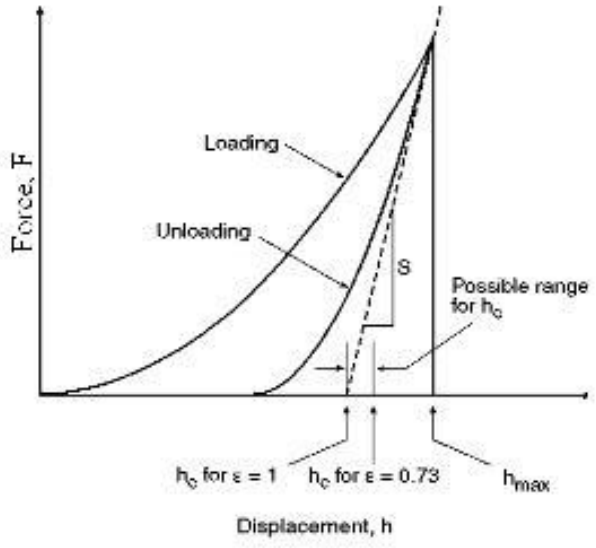


Figura 15: Imagen de la curva fuerza-desplazamiento al cargar y descargar obtenida en un ensayo de nano-indentación [18].

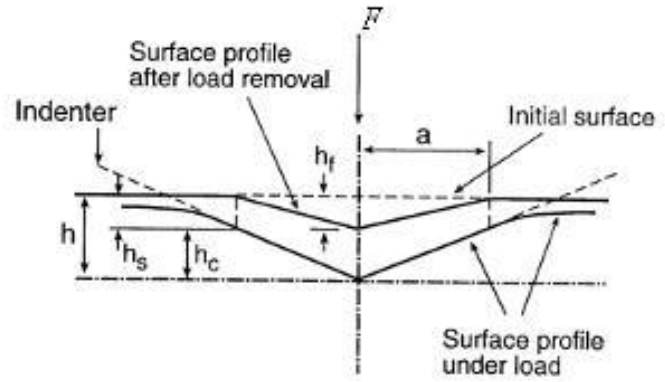


Figura 16: Imagen que representa el esquema de un ensayo de nano-indentación (la punta penetra una distancia h. Tras descargar y producirse la recuperación elástica la superficie vuelve retorna solo hasta h_s) [18].

Los resultados obtenidos se sometieron entonces a un ajuste gaussiano, y los resultados obtenidos para el módulo de Young en dirección longitudinal fueron de un valor medio de 11,51Gpa con una desviación típica de 1.84. Los valores del coeficiente de Poisson fueron obtenidos mediante mediciones extensiométricas en muestras de hueso. El valor del coeficiente de Poisson obtenido fue de un valor medio de 0.17 y una desviación típica de 0.06.

Para la línea cementante se tomaron valores del módulo de Young un 25 por ciento inferiores a los obtenidos para las osteonas y su coeficiente de Poisson se asume de un valor de 0,49 (Sabelman *et al*., 1997 Jul 20-23).

Por último para la matriz intersticial, se usaron valores un 10 por ciento mayores que el de las osteonas para el módulo de Young, unos 12,66Gpa y un 10 menores para el coeficiente de Poisson, 0,153, conforme a los resultados obtenidos mediante nano-indentación por Rho *et al* (2002).

Tabla resumen de las propiedades de las distintas fases que constituyen el modelo [4].

fase	Módulo de Young(Gpa)	Coeficiente de Poisson
Osteona	11,51	0,17
Matriz intersticial	8,63	0,49
Línea cementante	12,66	0,153

3.3 Análisis del modelo

Una vez conocidos los valores de las dimensiones y propiedades mecánicas de los distintos componentes del modelo se procederá a definir el análisis que será llevado a cabo.

El tipo de análisis llevado a cabo es un análisis cuasiestático, al suponerse que las cargas varían de un modo suficientemente lento como para que no se generen fuerzas inerciales, de este modo se considera que la respuesta mecánica no varía en función del tiempo. Se considera también, como se ha mencionado en el apartado anterior, que trabajaremos en un régimen elástico, es decir asumimos que no se superará el límite elástico de ninguno de los materiales presentes en el modelo y que en consecuencia, no se producirán deformaciones permanentes.

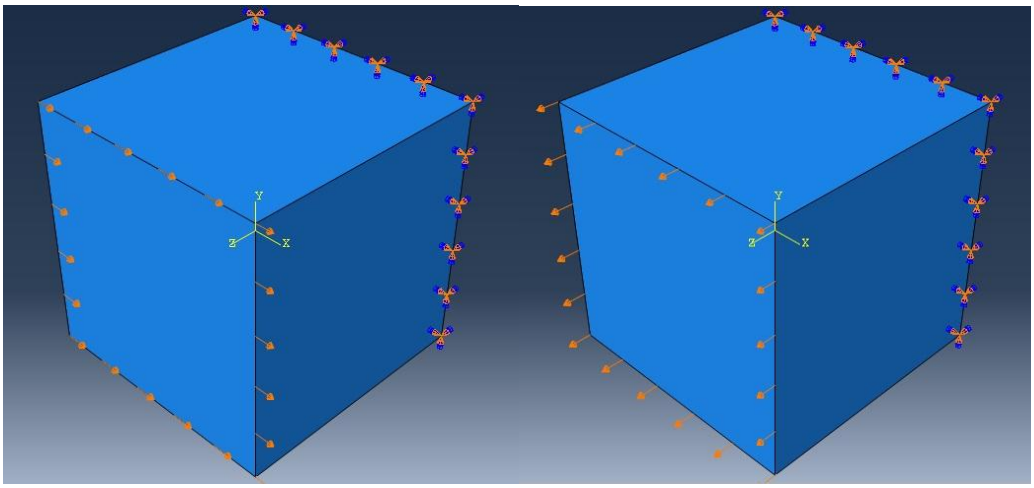
Como es el valor de numerosas constantes elásticas el que se quiere obtener, serán precisos implementar distintos casos casos de carga que nos permitan obtenerlos. Como norma general se definirá un empotramiento en una cara del cubo del volumen representativo, y en la cara opuesta se le aplicara el tipo de carga que corresponda. La carga se ha decidido que consistirá en imponer una deformación de 0,05 en la cara libre.

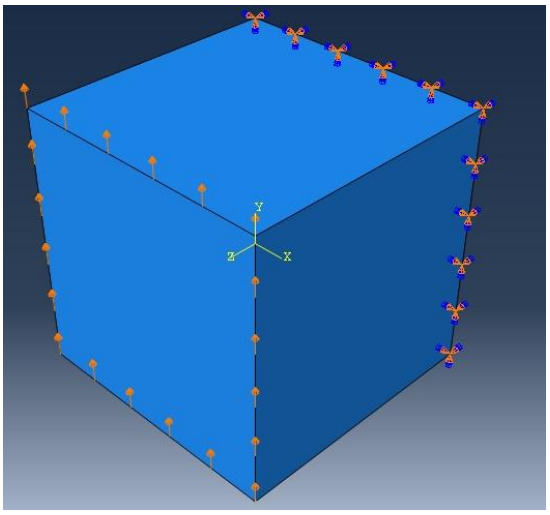
Para obtener el módulo de elasticidad en cierta dirección se impondrá el desplazamiento paralelo a esa dirección, y se obtendrán además en este caso de

carga los coeficientes de Poisson en relación a las otras dos direcciones. Para obtener los casos de cortadura el desplazamiento se aplica en dos situaciones distintas para las dos direcciones paralelas al plano del empotramiento, buscando obtener así los módulos de cortadura correspondientes. Por ejemplo para obtener el módulo de Young en dirección X y los coeficientes de Poisson XY y XZ el desplazamiento se impondría en el plano paralelo al plano YZ en dirección X. Para obtener los módulos de cortadura, en ese mismo plano se impondrían los desplazamientos en direcciones Y y Z.

Para obtener las constantes elásticas en las otras dos direcciones se llevaría a cabo un proceso análogo en dichas direcciones.

También se ha decidido finalmente que serán 5 los modelos que se construirán para así poder contrastar varios resultados y conseguir una mejor idea del comportamiento del hueso cortical.





Figuras 17-19: Imágenes que ejemplifican la distribución de desplazamientos en los distintos casos de carga para una dirección [obtención propia].

3.4 Herramientas informáticas a utilizar

Para la construcción del modelo y para poder realizar los cálculos se utiliza software especializado en computer aided design/engineering (CAD/CAE), en particular Abaqus CAE. Además nos ayudaremos del código de programación Python para automatizar la creación del modelo todo lo que sea posible, ya que son bastantes los modelos que habrá que implementar para realizar los análisis de malla y RVE y hasta llegar a los 5 modelos óptimos finales.

3.4.1 Abaqus 6.12

SIMULIA de Dassault Systemmes ofrece una potente suite completa con numerosas aplicaciones relacionadas con el mundo de la ingeniería industrial y especialmente mecánica, que permiten entre otras cosas, realizar análisis dinámicos (para simular impactos por ejemplo), simular estructuras marítimas, estudios de efectos térmicos o de dinámica de fluidos.

Para este proyecto hemos usado concretamente Abaqus CAE en su versión 6.12, que permite llevar a cabo todo el proceso de modelado y analizar modelos estructurales apoyándose en el método de los elementos finitos. Este programa permite importar modelos CAD de otros programas como CATIA o Pro/ENGINEER también ampliamente utilizados en este ámbito. También facilita el post-procesado de los datos obtenidos en el análisis permitiéndonos obtener gráficas o aplicar numerosas funciones a dichos datos. Tiene opción de modelar estructuras de 1,2 y hasta 3 dimensiones y de elegir entre diversos tipos de elementos finitos. Además permite usar un gran número de modelos de comportamiento mecánico tanto elásticos como plásticos. También permite simular movimientos e interacciones entre distintas partes o piezas, por lo que para proyectos de este tipo o de cálculo estructural en general supone una de las opciones más completas y atractivas [21,22].

3.4.2 Python

Python es un lenguaje de programación de uso general ampliamente utilizado hoy en día nacido a finales de los años 80 como sucesor del lenguaje ABC. Es un lenguaje caracterizado por su simplicidad sintáctica, orientado a tener una más fácil lectura y usar un menor número de líneas de código que otros lenguajes como C++. Comandos como if o for no necesitan signos de puntuación específicos sino tan solo usar sangría en el código que deben loopear.

En este proyecto se ha usado en conjunto con Abaqus, mediante scripts, que son archivos de texto que contienen un código que le dice a Abaqus lo que debe de hacer o crear. Cada vez que se crea un modelo en Abaqus, se genera un archivo que contiene todo el código del programa necesario para crearlo. El objetivo de usar Python era el de no tener que repetir cada modelo una y otra vez ni tener que empezar desde cero cada vez que variase la estructura, además de automatizar tareas repetitivas o pasos que deban realizarse en todos los modelos como la creación de un step o del módulo assembly.

Python nos permitió programar distintos scripts que se encargasen por ejemplo de generar la distribución de las distintas osteonas en el cubo del volumen

representativo, asignar las distintas propiedades elásticas a cada una de las fases o definir y realizar el mallado final de la estructura, permitiéndonos ahorrar una gran cantidad de tiempo y facilitando notablemente el estudio del modelo[25,28].

3.5 El método de Elementos Finitos

El método de los elementos será la herramienta básica de este proyecto. El Método de los Elementos Finitos es un método numérico desarrollado en los años 60 que aporta una eficiente aproximación para la resolución de problemas continuos.

Su planteamiento básico consiste en, como es muy difícil obtener las ecuaciones algebraicas que describan el comportamiento de cierto dominio, este se divide en elementos más pequeños y simples, cuyas ecuaciones algebraicas sean más fáciles de obtener. Esta división se conoce como “discretización” y da lugar a una malla de elementos. El comportamiento de estos elementos en que se subdivide el medio continuo se obtiene a partir de los parámetros asociados a los puntos que delimitan cada elemento, conocidos como nodos.

El sistema matemático del modelo discretizado se obtiene mediante el ensamblaje de los distintos elementos, siendo las incógnitas del sistema los valores de los parámetros en los nodos. El comportamiento de puntos distintos a los nodos, localizados en el interior de los elementos se halla a partir del valor en los nodos utilizando las respectivas funciones de interpolación o de forma. Esto repercute dando lugar a uno de los pocos inconvenientes que presenta este método, y es que al usar funciones de interpolación, los resultados serán más precisos cuanto mayor sea el grado del polinomio usado, pero nunca llegaran a ser los valores exactos, por lo que es preciso ser capaz de interpretar los resultados. Esto también origina que factores como el número de nodos y la densidad de malla den lugar a variabilidad en los resultados, o que si los elementos son de un tamaño demasiado elevado, se produzca una distorsión demasiado grande y los resultados en el interior de los elementos difieran demasiado de los reales.

Desde el punto de vista del cálculo estructural el método de los elementos finitos se torna útil permitiendo analizar estructuras de geometrías muy complejas o como en nuestro caso, de muy pequeño tamaño. El método de los elementos finitos se considera una generalización del cálculo matricial en lo referente a estructuras.

La forma básica que tendrá el método aplicado a una estructura es la del sistema K*D = F, donde K es la matriz de rigidez de la estructura, D es un vector que contiene los desplazamientos y giros de cada uno de los nodos de la estructura y F otro vector que incluye las fuerzas y reacciones en los nodos de la estructura. Este sistema se obtiene apoyándose en el método de Rayleigh-Ritz, que permite obtener una solución aproximada al campo de desplazamientos del sólido elástico, a partir de la aplicación del principio de la energía potencial total mínima. Este teorema afirma que para un sólido elástico, de todos los campos de desplazamiento posibles, aquellos correspondientes al equilibrio harán extremo el valor de la energía potencial. Si además el extremo resulta ser un mínimo, el equilibrio será estable. Finalmente, con las condiciones de contorno de que se dispongan, el sistema matricial se reduce y se obtienen los desplazamientos y reacciones oportunas.

Para sistemas matriciales de órdenes elevados el cálculo a mano resulta difícil de llevar a cabo a efectos prácticos, de ahí que con la tecnología disponible hoy día, sofisticados softwares informáticos hayan sido desarrollados para llevar a cabo estos cálculos.

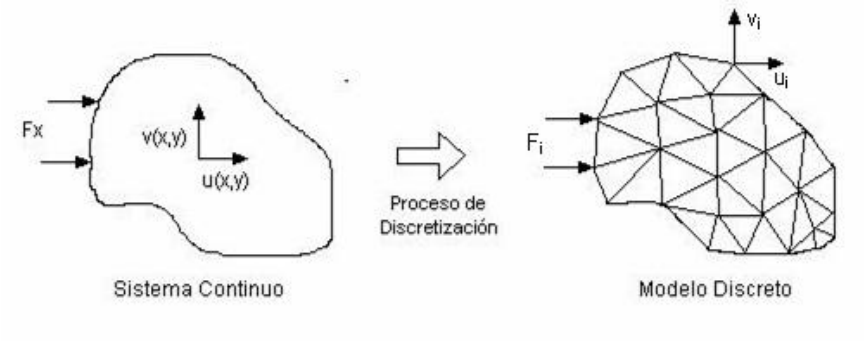


Figura 20: Imagen que representa la discretización de un sistema continúo dividiéndolo en elementos más simples [36].

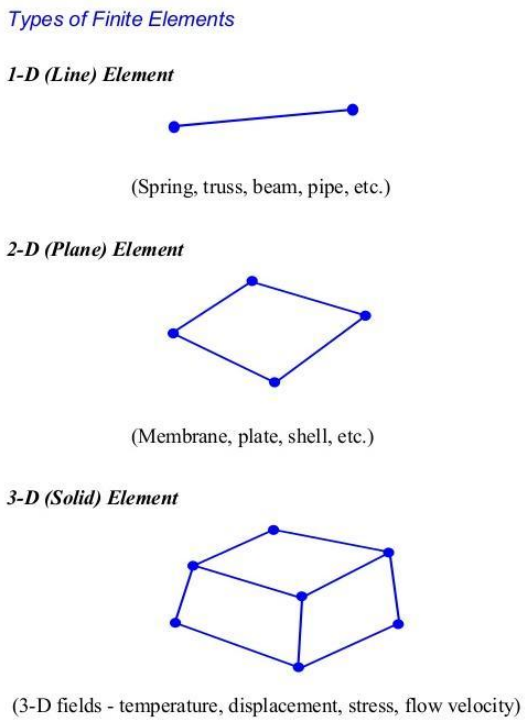
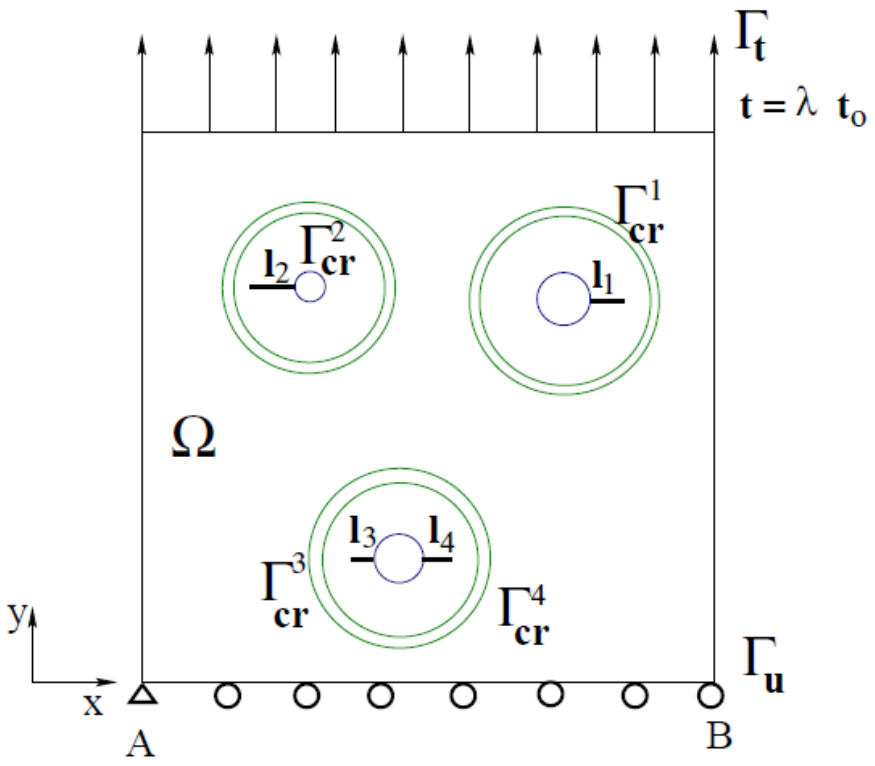


Figura 21: Imagen que ilustra los distintos tipos de elementos finitos en función de las dimensiones que comprendan.

CAPITULO IV: CREACIÓN DEL MODELO NUMÉRICO



En este capítulo se describirá paso a paso todo el proceso que se ha llevado a cabo para crear el modelo numérico que represente el volumen representativo de tejido cortical y dejarlo preparado para analizarlo y obtener así los distintos resultados requeridos. Se incluirán pasos llevados a cabo en Abaqus y/o en un script de Python explicando la forma en que interactúan entre sí. El script de Python fue editado utilizando el programa wordPad

En este capítulo solo se indicarán los pasos necesarios para uno de los 5 modelos finales, pero ha de tenerse en cuenta que fueron numerosas pruebas e intentos los que se realizaron: se empezó creando tan solo un cubo macizo sin osteona alguna, después se buscó la forma de introducir las osteonas y finalmente se impuso la condición de que no hubiese contacto entre osteonas.

Ha de ser sabido también que para crear el modelo se usan un total de 3 scripts distintos cada uno con determinadas funciones, ya que ciertos pasos resultaba más fácil y/o eficiente realizarlos directamente desde la interfaz de Abaqus.

4.1 Inicialización de un nuevo modelo

La inicialización de un modelo nuevo en Abaqus ha sido totalmente programada y forma parte del primer script. Ha de tenerse en cuenta al observar el código programado que el script lee y ejecuta el código de arriba abajo y que las líneas con un # son comentarios y simplemente sirven como aclaración o guía.

La primera parte del código es la siguiente:

```
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import random
import regionToolset
import math
```

Figura 22: Script 1.1 [obtención propia].

Aunque este código no ejecuta ningún comando aparentemente en Abaqus, es imprescindible para que el script se ejecute, ya que importa todas las bibliotecas de datos y funciones y da acceso al script a todos los objetos que estas incluyen. Todas las librerías excepto random y math están relacionadas con funcionalidades y objetos de Abaqus, mientras que random y math permiten el uso de determinadas funciones matemáticas.

La siguiente parte del código se encarga de obtener y definir varias constantes que serán de gran importancia para determinar varios parámetros del modelo:

```
#model constants

size=getInputs(fields = (('lado del cubo', '0.001'),
                ('fracción de volumen de osteonas', '0.65')),
                     label= 'Introduzca los siguientes
parametros',
                     dialogTitle='tamaño')

#variables principales
lsq=float(size[0])
fmax=float(size[1])
Amax = fmax*lsq*lsq
Aost = 0
pi = 3.14
```

Figura 23: Script 1.2 [obtención propia].

En esta parte del código, se empieza creando un vector de dos componentes llamado “size” que contiene información sobre las medidas del modelo. Estos valores se obtienen mediante la función *getInputs, que* hace aparecer una ventana en pantalla para que el usuario introduzca los datos que desee. Cada uno de los espacios a rellenar va acompañado de una pequeña descripción y además se pueden incluir valores por defecto ya sea para guiar al usuario o bien porque son los valores más utilizados. Concretamente de este vector se obtiene el valor del tamaño de lado del cubo del modelo (“lsq”) y la fracción de volumen que deberán ocupar las osteonas (“fmax”).

A continuación otras variables se obtienen, como “Amax” que hace referencia al área máxima que ocuparan las osteonas, multiplicando la fracción de volumen por el lado del cubo al cuadrado, mientras que otras se definen directamente como “Aost”, inicializada con valor “0”, que se utilizara como variable contador del área ocupado por las osteonas según se vayan creando, y “pi” que se utilizara en distintos cálculos.

La creación del archivo del modelo en Abaqus en sí se consigue mediente el siguiente bloque:

```
#modelo
sample = mdb.models['Model-1']
```

Figura 24: Script 1.3 [obtención propia].

Así se define el modelo, llamado “sample”, lo que nos permitirá referirnos al modelo en un futuro de una forma mucho más simple.

4.2 Creación del cubo base, módulo part

Una vez definidas las constantes anteriores se puede proceder a crear el cubo macizo que albergará las osteonas, programando distintas acciones que serán

llevadas a cabo dentro del módulo ‘part’ de Abaqus, destinado a crear y modificar la geometría que se precise:

```
#parte base
perfilsq = sample.ConstrainedSketch(name='perfil', sheetSize=1.0)
perfilsq.rectangle(point1=(0.0, 0.0), point2=(
    lsq, lsq))
cube = sample.Part(dimensionality=THREE_D, name='sample1', type=
    DEFORMABLE_BODY)
cube.BaseSolidExtrude(depth=lsq, sketch=
    perfilsq)
```

Figura 24: Script 1.3 [obtención propia].

Para crear una de las caras del cubo, se crea un objeto, denominado “perfilsq” mediante la función *ConstrainedSketch, un* nombre y el tamaño de la cuadricula han de ser especificados. Dentro del objeto creado se crea el perfil de la cara del cubo mediante el comando *rectangle,* el cual crea un cuadrado a partir de dos puntos introducidos, introduciendo en uno de ellos el valor del lado anteriormente obtenido.

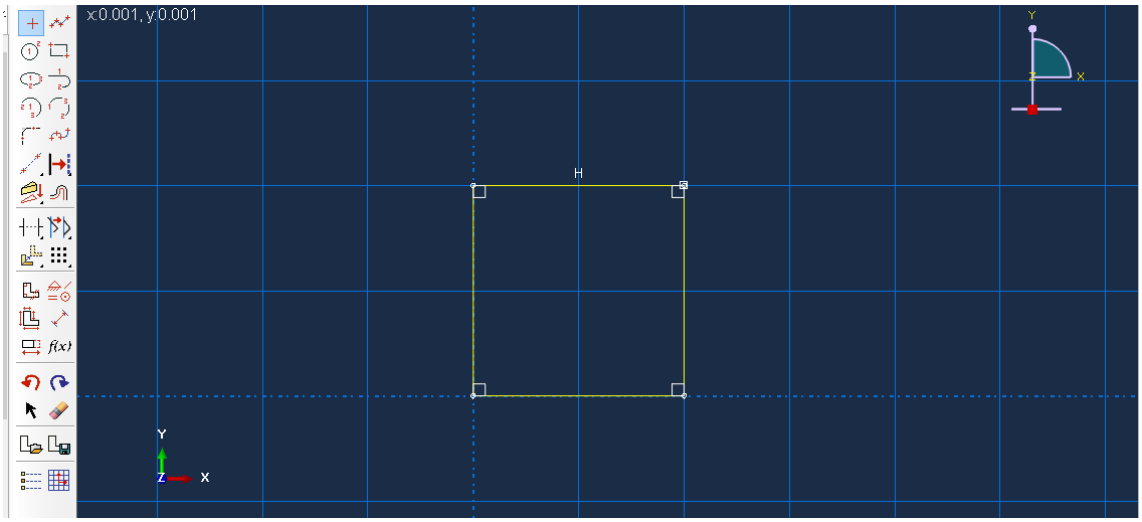


Figura 25: Imagen que muestra la interfaz para dibujar bocetos en Abaqus, ejemplificando la creación del perfil cuadrado [obtención propia].

El siguiente paso crea un objeto ‘part’ llamado “cube”. El primer parámetro necesario es un nombre, el segundo lo define como un objeto en 3 dimensiones y el último define su tipo como objeto deformable.

Por último se crea el cubo mediante el comando *BaseSolidExtrude,* que extiende el perfil elegido, la distancia que se quiera, en este caso se extiende el perfil cuadrado una distancia igual al lado del cubo.

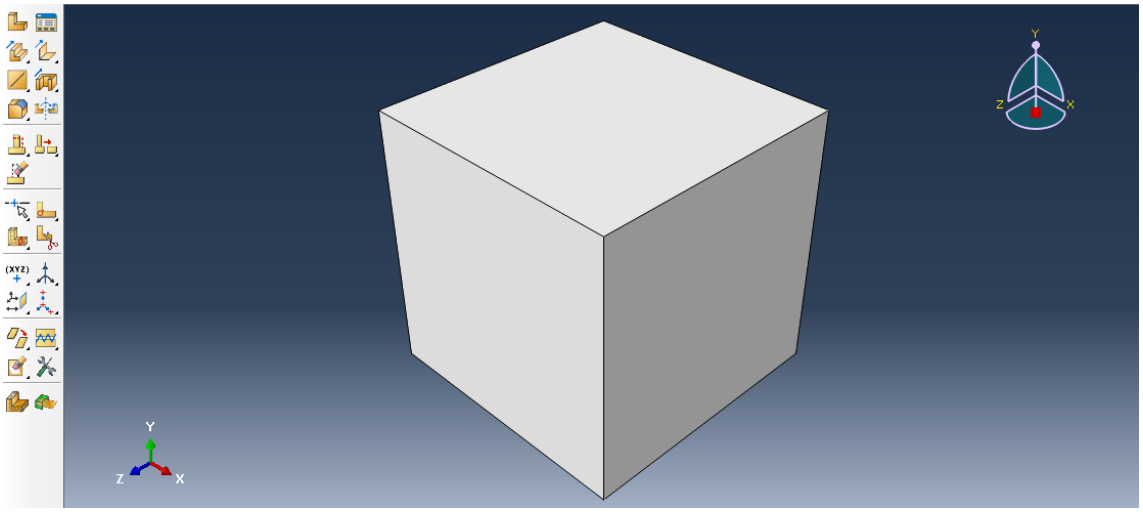


Figura 26: Imagen que muestra el cubo que sería creado ejecutando tan solo el código introducido hasta el momento [obtención propia].

A continuación un pequeño bloque de código auxiliar para crear un eje auxiliar que sirva para referencias de distintos comandos:

```
#datum axis
pdatum = (lsq, lsq, lsq/2)
datumedge = cube.edges \
                    .findAt(pdatum)
cube.DatumAxisByThruEdge(edge=datumedge)
```

Figura 27: Script 1.4 [obtención propia].

El primer paso es definir un punto del espacio especificando sus coordenadas, en este caso el punto está situado en la mitad de una de las aristas superiores del cubo. Para poder crear un eje refiriéndonos a la arista deseada, necesitamos crear este punto, ya que desconocemos el número de la lista que le ha asignado el programa, por lo que utilizamos el comando *findAt* que detecta y devuelve objetos situados en un punto determinado, o a una distancia de 1e-6 metros

como máximo. Una vez definido el eje, lo utilizamos para crear el eje auxiliar mediante el comando *DatumAxisByThruEdge.*

La creación de este eje auxiliar tiene sentido ya que en la creación de osteonas de forma aleatoria, partes del cubo que contengan un arista pueden ser eliminadas por los vaciados que requieren los canales de Havers.

A continuación se introduce el Código con los bucles necesarios para lograr la correcta distribución y creación de las osteonas con las características y dimensiones requeridas. Este código se ha subdividido en pequeñas porciones para entender mejor su funcionamiento:

```
#vaciados1
k = 1
xposition = [0]
yposition = [0]
radio = [0]
osteona = [0]
lcem = [0]
for i in range(200) :
    a = random.uniform(0,lsq)
    b = random.uniform(0,lsq)
    r = random.uniform(0,0.00006)
    c = a+r
    o = random.uniform(0,0.00018)
    l = random.uniform(0.0000015,0.00001)
    lu = c+o
    ld = lu+l
    lum = c+(o/2)
    ldm = lu+(l/2)
    tamano = r+o+l
    print i
```

Figura 28: Script 1.5 [obtención propia].

Para comenzar se crea una variable “k” que será utilizada para comparaciones de bucles y llevar una cuenta del número de osteonas creadas hasta el momento, se inicializa con un valor k = 1 para que la comparación funcione correctamente.

Se crean también varios vectores que nos serán de ayuda para implementar la condición de que las osteonas no estén en contacto.

La creación de osteonas comienza con un bucle *for,* que limita el número de intentos de crear una osteona a 200, para evitar que el bucle se alargue en exceso. Para cada valor de 'i' dentro de este bucle *for* se generan distintos valores que determinan la posición y tamaño de las osteonas, para la creación de estos valores se ha utilizado la función random.uniform que devuelve un número aleatorio tipo 'float' comprendido dentro de los valores límite que se establezcan:

"a" y "b" son dos variables con un valor límite del tamaño del lado del cubo, que servirán para determinar la posición del centro de cada osteona. "r" "o" y "l" determinan el radio del canal de Havers, y la extensión de cada osteona y su línea cementante. "c" es un punto auxiliar que será necesario para determinar el límite del canal de Havers. De forma análoga "lu" y "ld" representan una de las coordenadas exteriores de la osteona y la línea cementante respectivamente.

"lum" y "ldm" serán necesarias más adelante para poder locacalizar osteonas y líneas cementantes con el comando findAt, ya que llevar un control del número que ocuparía cada boceto en la lista resultaría difícil y tedioso.

La última variable aquí definida es "tamano", que se utiliza para aumentar el área acumulada hasta el momento tras la creación de cada osteona.

Por último se muestra al usuario el valor de 'i' para llevar un seguimiento y determinar cuántos intentos han sido necesarios.

La siguiente parte del código está comprendida dentro del bucle *for*:

```
#vaciados2
    if Aost < Amax :

      for j in range(k) :
         diametros = tamano+radio[j]+osteona[j]+lcem[j]
         distcentro =math.sqrt((a-xposition[j])*(a-xposition[j])
         +(b-yposition[j])*(b-yposition[j]))
         print diametros
         print distcentro
         if diametros > distcentro :
            condicion = 'error'
            break
         else:
            condicion = 'true'
```

Figura 29: Script 1.6 [obtención propia].

En esta parte un bucle *if* se utiliza para imponer la condición de que la máxima fracción de volumen que ocupan las osteonas sea del 65 por ciento. Como se trata de un cubo y la extensión de este coincide con la de las osteonas, se han simplificado los cálculos de forma que, es el área lo que se compara. En este caso si el área acumulada hasta el momento "Aost" es menor que la máxima permitida "Amax" el bucle continua.

Entraría entonces en otro bucle *for* en conjunto con el comando *in range()*, con este bucle se comparan los parámetros de una nueva osteona con los de todas las creadas anteriormente: "diámetros" es una variable que cuantifica la suma de los diámetros de la nueva osteona y la anterior correspondiente a la variable "j" en los distintos vectores de posición y dimensiones. "distcentro" utiliza el teorema de Pitágoras para calcular la distancia entre los centros de la nueva posible osteona y la correspondiente a "j" con la función *sqrt* (raíz cuadrada).

Se muestra en pantalla entonces el valor de estas variables como comprobación.

Se valora entonces con otro *if* si la distancia entre centros es mayor que la suma de diámetros correspondiente, la nueva osteona no estará en contacto con la osteona ya creada, si además se cumple esta condición al compararla con todas las demás osteonas, la nueva osteona podrá ser creada. Para conseguir esto se crea una variable "condición" que en caso de que haya contacto entre la nueva y alguna de las osteonas ya creadas adquiere el valor 'error' y el bucle para y vuelve al principio para intentar crear una nueva osteona con otros valores. Si no hay contacto, la variable "condición" toma el valor 'true' y el bucle continúa con el resto del código:

```
#vaciados3

      if condicion == 'true' :

         pcarav = (lsq/2, lsq/2, lsq)
         pladov = (lsq, lsq/2, lsq)
         carav = cube.faces \
                        .findAt(pcarav)
         ladov = cube.edges \
                        .findAt(pladov)
         perfilcrt = sample.ConstrainedSketch(gridSpacing=0.005,
            name='vaciado',
            sheetSize=0.4, transform=
            cube.MakeSketchTransform(
            sketchPlane=carav,
            sketchPlaneSide=SIDE1,
            sketchUpEdge=ladov,
            sketchOrientation=RIGHT, origin=(0.0, 0.0, lsq)))
         perfilcrt.sketchOptions.setValues(
            decimalPlaces=3)
         cube.projectReferencesOntoSketch(filter=
            COPLANAR_EDGES, sketch=perfilcrt)
         perfilcrt.CircleByCenterPerimeter(center=(
            a, b), point1=(c, b))
         cube.CutExtrude(flipExtrudeDirection=OFF,
            sketch=perfilcrt, sketchOrientation=
            RIGHT, sketchPlane=carav,
            sketchPlaneSide=SIDE1, sketchUpEdge=
            ladov)
```

Figura 30: Script 1.7 [obtención propia].

Una vez “condición” ha obtenido el valor ‘true’, un nuevo *if* es validado y se procede a crear el perfil de la nueva osteona.

Se definen puntos para identificar mediante *findAt* la cara y la arista usadas como referencia para donde crear los bocetos de las osteonas (para todas las osteonas la cara son las mismas, pero no sabemos qué número son dentro de la lista generada por el programa).

Se define un nuevo perfil llamado “perfilcrt” con el mismo método que anteriormente, utilizando además el método *MakeSketchTransform* que nos permite crearlo en la ‘part’ ya creada y que nos pide introducir la cara donde se creara, el lado de la cara, la arista de referencia, la orientación de la cuadricula y el origen de esta. El comando *projectReferencesOntoSketch* hace posible seleccionar referencias ya creadas en el cubo.

CircleByCenterPerimeter crea el círculo que representa el canal de Havers, usando las variables “a” y “b” como centro y “c” y “b” para definir un punto del círculo. Para crear el canal de Havers se usa el método *CutExtrude* y se especifica que utilice el perfil recién creado y las referencias necesarias.

Se definen a continuación varios puntos que deben usarse para referenciar caras y aristas que deben usarse más adelante:

```
#perfiles particiones1
pcarao = (lsq/2, lsq/2, lsq)
pladoo = (lsq, lsq/2, lsq)
carao = cube.faces \
                .findAt(pcarao)

ladoo = cube.edges \
                .findAt(pladoo)
```

Figura 31: Script 1.8 [obtención propia].

Al igual que otras veces primero se define un punto mediante sus coordenadas, para después utilizar el método *findAt.*

El código a continuación constituye la última parte del código necesario para crear las osteonas y su distribución:

```
#perfiles particiones2
          perfilOst = sample.ConstrainedSketch(gridSpacing=
             0.005,name='osteona',
             sheetSize=0.4, transform=
             cube.MakeSketchTransform(
             sketchPlane=carao,
             sketchPlaneSide=SIDE1,
             sketchUpEdge=ladoo,
             sketchOrientation=RIGHT, origin=(0.0, 0.0, lsq)))
          perfilOst.sketchOptions.setValues(
             decimalPlaces=3)
          cube.projectReferencesOntoSketch(filter=
             COPLANAR_EDGES, sketch=perfilOst)
          perfilOst.CircleByCenterPerimeter(center=(
             a, b), point1=(lu, b))
          perfilOst.CircleByCenterPerimeter(center=(
             a, b), point1=(ld, b))
          cube.PartitionFaceBySketch(faces=
             carao, sketch=perfilOst, sketchUpEdge=
             ladoo)
          Aost = Aost+pi*tamano*tamano
          xposition.append(a)
          yposition.append(b)
          radio.append(r)
          osteona.append(o)
          lcem.append(l)
          k = k+1
print k
```

Figura 32: Script 1.9 [obtención propia].

Se crea otro perfil llamado “perfilOst”, y al igual que en el anterior es necesario utilizar el comando *MakeSketchTransform* para poder modificar la ‘part’.

En este caso se crean dos nuevos círculos que delimitan la osteona y la línea cementante respectivamente, siendo necesarias las variables que definan su centro y un punto del propio círculo, dando su uso correspondiente a las variables “lu” y “ld”. Una vez definidos estos dos círculos, se utilizan para crear una división en la cara correspondiente mediante el comando *PartitionBySketch*.

En este punto el código ya se ha hecho todo lo que debe por su parte para crear una nueva osteona. Con los datos de la nueva osteona se actualiza el valor del área ocupada por osteonas, y se añaden los distintos valores de posición y dimensiones a sus vectores correspondientes para poder compararlos al crear una nueva osteona mediante el comando append que los añade al final del vector especificado y aumenta su tamaño en un componente. Por último se incrementa el valor de “k” para tener en cuenta la nueva osteona en el bucle

for+in range. Se muestra en pantalla el valor de “k” una vez completado el proceso el cual nos indica el número de osteonas total que ha sido creado.

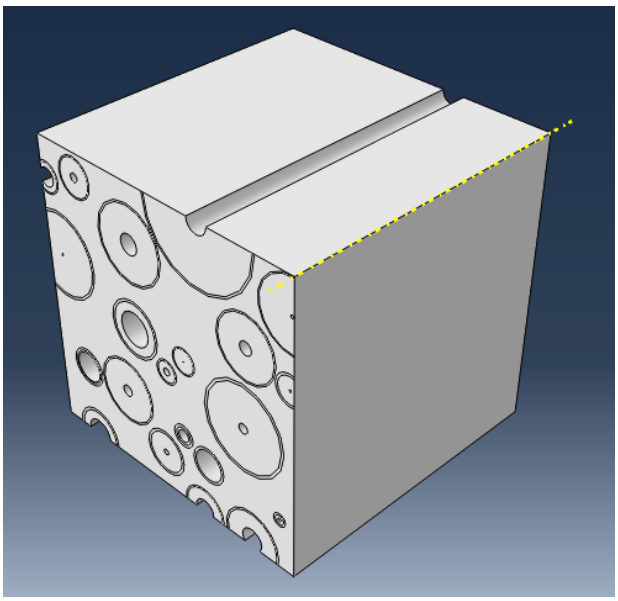


Figura 33: Imagen que ilustra el aspecto del modelo una vez ejecutado todo el código hasta el momento [obtención propia].

Como se puede observar en el dibujo, los perfiles de las osteonas están todos creados, pero solo los canales de Havers extan extendidos hasta la cara opuesta. Esto se debe a que en la creación de los perfiles, como estamos trabajando con dimensiones muy pequeñas y en algunos casos las osteonas intersectan con las aristas del cubo, en ocasiones las líneas presentaban subdivisiones inesperadas, por lo que por simplicidad se decidió que la acción de extender los perfiles hasta la cara opuesta se realizaría manualmente directamente desde el programa. Para ello se utilizó la herramienta para crear particiones, eligiendo tipo ‘cell’ y el método ‘Extrude/Sweep edges’.

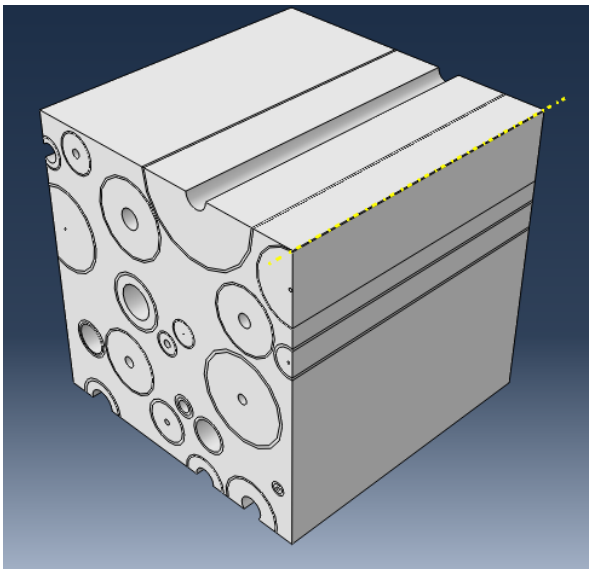


Figura 34: Imagen del aspecto del modelo una vez creadas todas particiones [obtención propia].

4.3 Asignación de materiales, módulo 'material'

La siguiente parte del script está asociada completamente al módulo 'material' de Abaqus. Su objetivo es el de crear los distintos materiales a utilizar y asignarles sus propiedades correspondientes:

```
#crear materiales
Mhueso = sample.Material(name='hueso')
Mhueso.Elastic(table=((12660000000.0, 0.153),
    ))
Most = sample.Material(name='osteona')
Most.Elastic(table=((11510000000.0,
    0.17), ))
Mlincem = sample.Material(name='lineacementante')
Mlincem.Elastic(table=((
    8632500000.0, 0.49), ))
```

Figura 35: Script 1.10 [obtención propia].

El primer material creado corresponde con la matriz intersticial. Se crea una un objeto llamado “Mhueso” para el material llamado ‘hueso’ y se introducen las propiedades descritas en el capítulo anterior (que recordamos definían a los distintos materiales como elásticos e isótropos) mediante el método *Elastic.* Los valores del módulo de Elasticidad y el coeficiente de Poisson son los requeridos.

Se crean de la misma manera los otros materiales ‘osteona’ y ‘lineacementante’ cada uno con sus propiedades correspondientes.

Una vez creados los materiales se puede proceder a crear sus secciones correspondientes:

```
#crea secciones

Shueso = sample.HomogeneousSolidSection(material='hueso',
name='hueso',
    thickness=None)
Sost = sample.HomogeneousSolidSection(material='osteona', name=
    'osteona', thickness=None)
Slincem =
sample.HomogeneousSolidSection(material='lineacementante', name=
    'lineacementante', thickness=None)
```

Figura 36: Script 1.11 [obtención propia].

Se va a crear una sección para cada uno de los materiales anteriores. Para crear cada una de ellas, al tratarse de una ‘part’ en 3 dimensiones, la sección ha de crearse mediante el método *HomogeneousSolidSection.* Especificar que material se le asigna, Introducir un nombre y un espesor es todo lo que se requiere, dando lugar a las 3 secciones ‘hueso’, ‘osteona’ y ‘lineacementante’ en concordancia con los materiales que se les asignan.

Todo el código anterior forma parte del primer script. La interrupción se debe a que es al ejecutar todo lo definido hasta ahora cuando se procede a extender las secciones de las osteonas manualmente para crear las particiones.

La asignación de secciones a cada una de las divisiones o particiones del modelo se hace también mediante un script, formando ya parte del segundo script del modelo:

```
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import random
import regionToolset
import math
```

Figura 37: Script 2.1 [obtención propia].

Este bloque de código es el mismo que en el script anterior, necesario para cada nuevo script creado.

La parte del código con el bucle que asigna las secciones comienza con el siguiente bloque:

```
#asignar secciones
s = 1
while s < k :
      xpos = xposition[s]+radio[s]+(osteona[s]/2)
      ypos = yposition[s]
      xpos2 = xposition[s]+radio[s]+osteona[s]+(lcem[s]/2)
      psost = (xpos, ypos, lsq)
      ostcell = cube.cells.findAt((psost,) )
      ostRegion = (ostcell,)
      cube.SectionAssignment(offset=0.0,
          offsetField='', offsetType=MIDDLE_SURFACE,
          region=ostRegion, sectionName='osteona',
          thicknessAssignment=
          FROM_SECTION)
```

Figura 38: Script 2.2 [obtención propia].

En primer lugar se crea una variable “s” que se utiliza para comparar en el bucle *while* a continuación, que se ejecutará siempre que “s” sea menor que “k” (número total de osteonas) y repitiendo así este bucle tantas veces como sea necesario para que todas las osteonas y líneas cementantes tengan asignada su sección.

Se definen 3 variables que nos ayudan a localizar las osteonas al usar el método *findAt*: para la posición ‘Y’ se toma el mismo valor que el del centro de la osteona, mientras que para la posición ‘X’ se toman dos valores que situan los puntos en un punto entremedias del canal de havers y el diametro exterior de la osteona (posición ‘X’ de la osteona, más el radio de su canal de Havers, más la dimensión de la osteona entre dos para localizar la osteona) y entre el límite de la osteona y el diametro externo de la línea cementante (posición ‘X’ de la osteona, más el radio de su canal de Havers, más la dimensión de la osteona, más la dimensión de su línea cementante entre dos para localizar la línea cementante).

Para asignar la sección correspondiente a una osteona se define un punto situado en la cara donde se encuentra el comienzo de esta, con la posición anteriormente explicada. Con este punto se selecciona la partición correspondiente y esta se asigna a una región necesaria para el siguiente comando.

La asignación en sí, se realiza mediante el comando SectionAssignment, al que se le especifica cuál de las 3 secciones debe ser asignada y se le indica la región definida arriba.

La asignación de la sección a una línea cementante se consigue con el siguiente bloque de código:

```
#asignar secciones2
      pslincem = (xpos2, ypos, lsq)
      lincemcell = cube.cells.findAt((pslincem,) )
      lincemRegion = (lincemcell,)
      cube.SectionAssignment(offset=0.0,
          offsetField='', offsetType=MIDDLE_SURFACE,
          region=lincemRegion,
          sectionName='lineacementante',
          thicknessAssignment=
          FROM_SECTION)
      s = s+1
pcell = (lsq/2, lsq/2, lsq/2)
cubecell = cube.cells.findAt((pcell,) )
cuboRegion = (cubecell,)
cube.SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE,
    region=cuboRegion, sectionName='hueso', thicknessAssignment=
    FROM_SECTION)
```

Figura 39: Script 2.3 [obtención propia].

La sección se asigna del mismo modo que para la osteona, pero definiendo otra región distinta, con otra partición seleccionada con el otro punto que aquí se define y especificando la sección 'lineacementante'.

Llegados a este punto se incrementa en 1 el valor de la variable "s" y el bucle vuelve al principio.

Una vez ejecutado todo el bucle solo falta asignar la sección a la parte correspondiente a la matriz intersticial (como solo es una no hace falta incluirla en un bucle). Para ello se sigue un proceso análogo: Se define un punto, localizado esta vez en el centro del cubo, con el que se encuentra la partición correspondiente a la matriz, la cual se asigna a una región necesaria para el comando *SectionAssignment.*

Si bien es cierto que esta parte del código que asigna las secciones suele funcionar, también es cierto que a veces genera errores, como por ejemplo en el caso de que una de las osteonas pase por el centro del cubo, la sección de la matriz intersticial no será asignada correctamente, y la osteona puede en ese caso tener dos osteonas asignadas. También pueden derivarse errores al asignar las secciones a las líneas cementantes, debido a fallos con el comando *findAt* ya que las dimensiones de las líneas cementantes son muy reducidas.

Es por esto por lo que una vez ejecutado este código se comprueba que todas las secciones estén correctamente asignadas (se puede verificar visualmente en el modelo ya que todo el cubo ha de estar en verde sin partes grises (sin sección asignada) ni amarillas (con más de una sección asignada)). Si se encuentra algún tipo de error resulta muy fácil corregirlo manualmente desde el propio Abaqus.

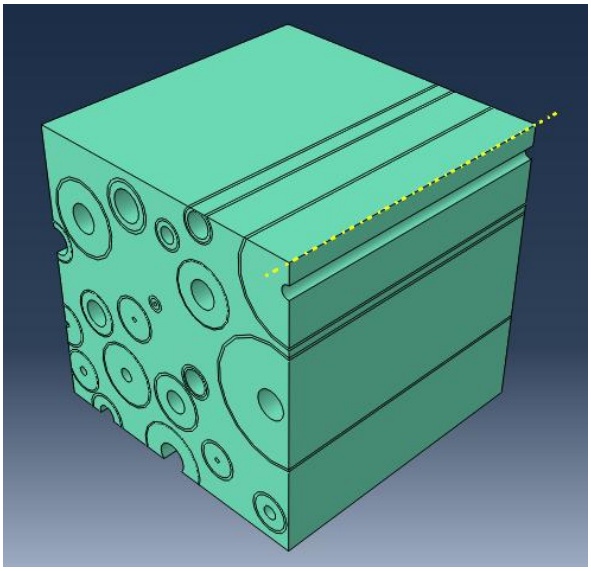


Figura 40: Imagen del aspecto que debería presentar el modelo una vez asignadas todas secciones correctamente [obtención propia].

4.4 Módulo 'assembly' y módulo 'step'

La siguiente parte del código se encarga de todo lo necesario realizar en el módulo 'assembly':

En este módulo se crean 'instances'

```
#assembly
cubeAssembly = sample.rootAssembly
cubeAssembly.DatumCsysByDefault(CARTESIAN)
cubeInstance = cubeAssembly.Instance(dependent=OFF,
name='sample1-1',
    part=cube)
```

Figura 41: Script 2.4 [obtención propia].

En este módulo se crean 'instances' sobre las cuales se podrá realizar el mallado. También es el utilizado para ensamblar varias 'part' entre sí y definir sus posiciones relativas cuando es necesario. Como en este caso solo hay una simplemente se crea una 'instance' a partir de la 'part' del cubo.

En primer lugar se define la variable "cubeAssembly" a la que se le asigna el 'rootAssembly' que es creado por defecto por el programa.

El comando DatumCsysByDefault establece el sistema de coordenadas cartesianas por defecto.

Se define ahora la variable "cubeInstance" que mediante el metodo *Instance* crea la 'instance' a partir del cubo con as osteonas, le asigna un nombre y en este caso lo define como independiente respecto a otras partes.

El siguiente módulo es el módulo 'step' y el código asociado a él es el siguiente:

```
#step
sample.StaticStep(name='Step-1', nlgeom=ON, previous='Initial')
```

Figura 42: Script 2.5 [obtención propia].

En este módulo se crean una serie de pasos representando cada uno un estado de carga. En este caso la carga solo será una por lo que solo hay crear un nuevo 'step', a parte del creado por defecto.

Se crea el 'step' mediante el método *StaticStep* ya que el análisis realizado es estático, se le asigna un nombre, se activa la geometría no lineal para avisar al programa de su presencia y se especifica el 'step' anterior al que se va a crear.

Esta es la última parte del segundo script, y desde este punto varias de las acciones necesarias a continuación se ejecutan manualmente.

En este módulo también se eligen las variables de las cuales se desea que el programa analice, calcule y muestre los resultados. Para poder decidir lo que se incluye en el análisis en cada momento, no se programó esta elección. Concretamente para este modelo y los resultados que se persigue conseguir se incluirán en el análisis las fuerzas y los desplazamientos.

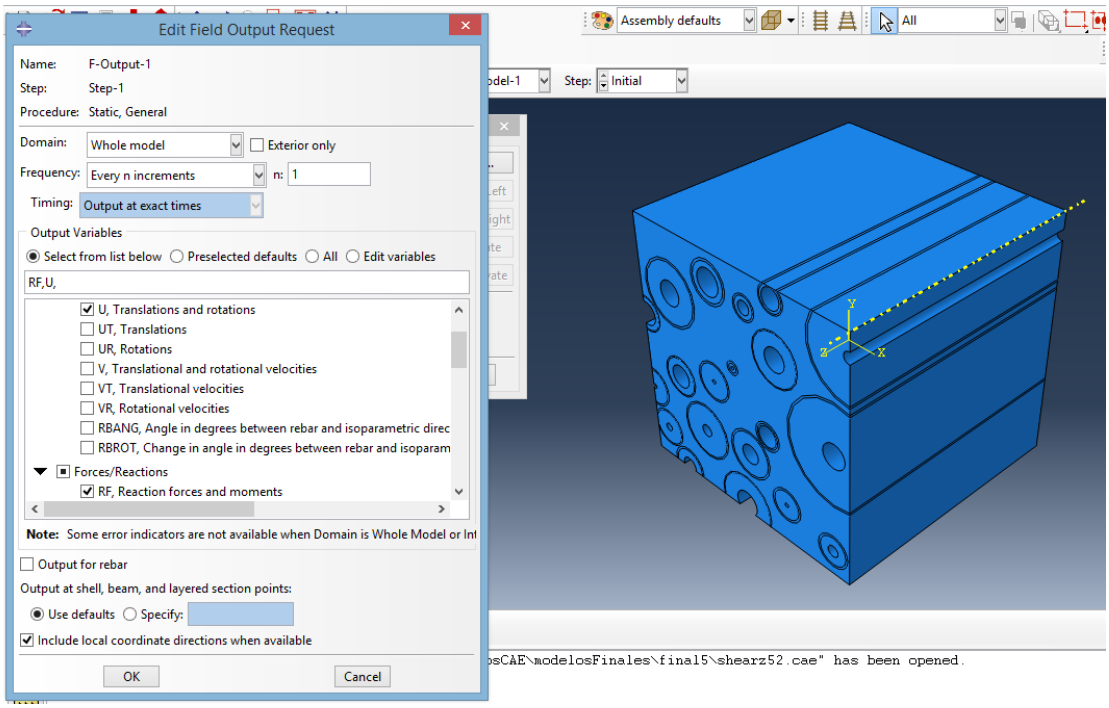


Figura 43: Imagen del módulo 'step' en que se observa la ventana de selección de variables que se pueden elegir [obtención propia].

4.5 Definición de los estados de carga, módulo 'load'

Este módulo es en el que se lleva a cabo todo lo relacionado con la definición de cargas y condiciones de contorno. Se ha elegido realizar totalmente a mano ya que son muchos los casos de carga a analizar y resulta muy sencillo cambiarlo en el momento y definir el estado que se quiera entonces.

Para cada uno de los casos se crea siempre un empotramiento en el 'step' inicial en la cara correspondiente, que se propaga por los demás 'step' a continuación del inicial. En Abaqus esto se realiza mediante la funcionalidad 'Create Boundary Condition', seleccionando 'encastre' y especificando la cara correspondiente del cubo.

En el siguiente ‘step’, el creado en el script, se define la carga respectiva, también mediante ‘Create Boundary Condition’ seleccionando esta vez ‘Displacement’, y especificando la dirección y el valor del mismo (5% del valor del lado del cubo).

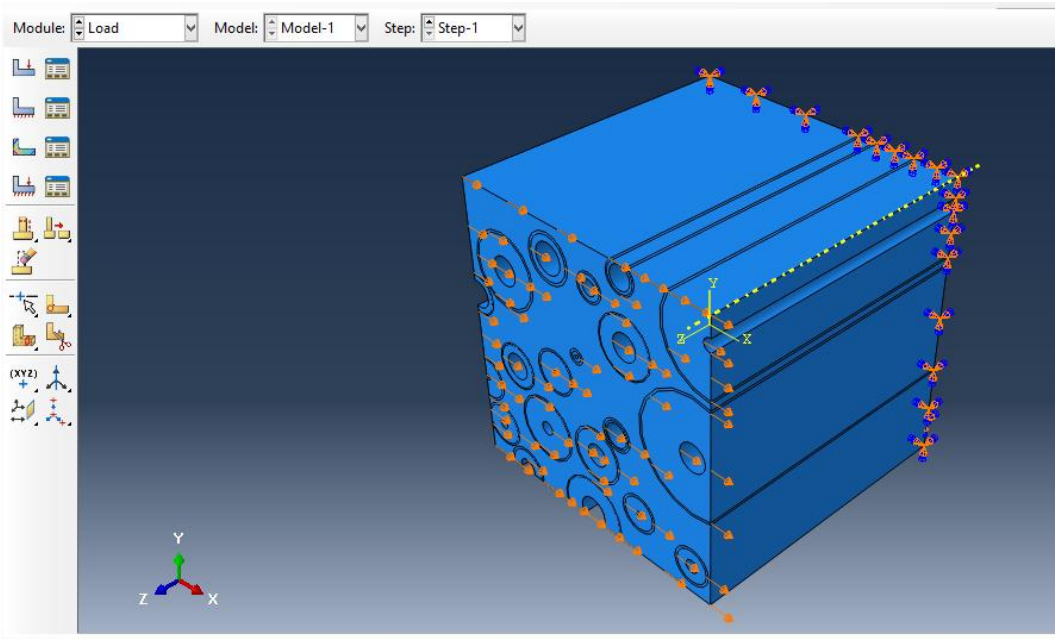


Figura 44: Imagen del módulo ‘load’ representando uno de los estados de carga completos del modelo [obtención propia].

4.6 Generación de la malla, módulo ‘mesh’

En este módulo del programa se realizan todas las acciones necesarias para discretizar el modelo, es decir, dar lugar a la malla de elementos finitos a partir de los cuales se obtendrán los resultados. Para ello se utilizó un tercer script que facilitase la creación de esta malla:

```
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import random
import regionToolset
import math
```

Figura 45: Script 3.1 [obtención propia].

```
#mesh
elem=getInput('tamaño de los elementos:')
print elem

meshRegion = (cubeInstance.cells,)
cubeAssembly.setMeshControls(elemShape=TET, regions=
cubeInstance.cells.getSequenceFromMask(
    ('[#1f ]', ), ), technique=FREE)

cubeAssembly.setElementType(elemTypes=(ElemType(
    elemCode=C3D20R, elemLibrary=STANDARD),
ElemType(elemCode=C3D15,
    elemLibrary=STANDARD), ElemType(elemCode=C3D10,
elemLibrary=STANDARD)),
    regions=meshRegion)


cubeAssembly.seedPartInstance(deviationFactor=0.1,
    minSizeFactor=0.1, regions=(
    cubeInstance, ), size=elem)
cubeAssembly.generateMesh(regions=(
   cubeInstance, ))
```

Figura 46: Script 3.2 [obtención propia].

La primera parte del código es el ya comentado necesario para cualquier nuevo script.

El programa necesita que le especifiquemos el tamaño de los elementos que ha de crear, y para obtener este valor se ha utilizado el mismo método que para obtener el del lado del cubo: se ha creado una variable a la que se le asigna el valor mediante la función *getInput*, que lo extrae de la información introducida por el usuario. Para comprobar que el valor es correcto se utilizó la función *print* para mostrarlo en pantalla y saber cuál fue el último valor elegido (acción útil para la realización del análisis de malla).

La región a mallar se define como todas las partes presentes en el modelo ('cubeInstance.cells').

Mediante el método *SetMeshControl*s se indican varios parámetros que definirán las características de la malla creada: el tipo de elementos es tetraédricos, la región elegida es la anteriormente definida, la técnica elegida en relación a la distribución de elementos es libre puesto que la estructura tienen zonas de geometría poco lineal.

El comando *setElementTypes* define el tipo de elementos que utilizará el programa en la malla. Se especifica el código de los elementos a utilizar mediante un código que utiliza el programa y también se especifica que biblioteca utilizar ('STANDARD' o 'EXPLICIT'), que por el tipo de análisis en este caso se elige 'STANDARD'. Se especifica también a que región se le aplica este tipo de elementos.

El siguiente comando es seedPartInstance que crea las 'semillas' donde se localizan los nodos de los elementos en todo el modelo. Se requiere introducir factores de desviación, la región donde crear estos nodos y el tamaño de los elementos, en este caso el introducido en *getInput*.

Por último, con todos los parámetros que se necesita especificar se procede a generar la malla utilizando el método *generateMesh* y especificando la región en que ha de ser creada.

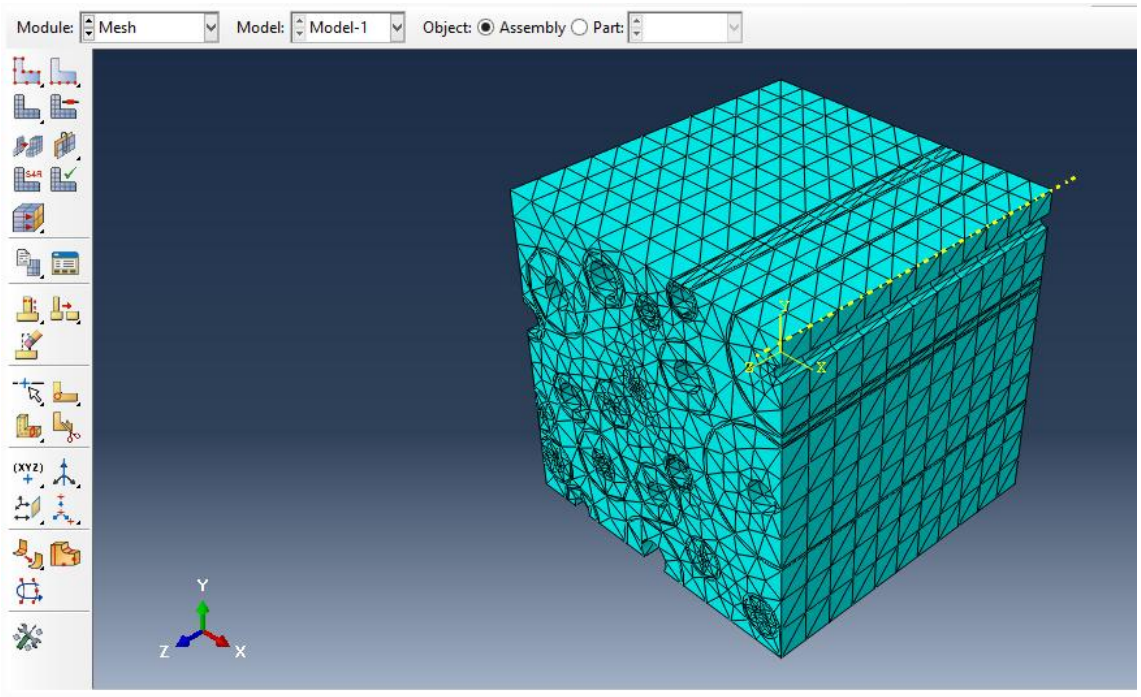


Figura 47: Imagen del módulo 'mesh' en el que se puede ver un ejemplo de un modelo ya mallado [obtención propia].

En ciertos casos de análisis de malla y/o RVE la malla no se generaba correctamente, por lo que manualmente para estos casos se utilizaba una herramienta de Abaqus llamada 'Virtual Topology' que creaba varias excepciones para ignorar 'entidades' demasiado pequeñas o de una geometría demasiado complicada a la hora de realizar el mallado.

4.7 Módulo 'job'

El módulo 'job' es aquel en que se crea un objeto que define el análisis en sí para todo el modelo y sistema de cargas creado hasta el momento. Este objeto 'job' se ejecuta y crea un archivo con los resultados del análisis correspondiente. También se puede monitorizar el análisis para que en caso de error el programa aborte la ejecución y nos informe de este error.

El código necesario en este módulo corresponde con la última parte del último script y es el siguiente:

```
#job

mdb.Job(atTime=None, contactPrint=OFF, description='',
echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True,
historyPrint=OFF,
    memory=90, memoryUnits=PERCENTAGE, model='Model-1',
modelPrint=OFF,
    multiprocessingMode=DEFAULT, name='cubeJob',
nodalOutputPrecision=SINGLE,
    numCpus=1, numGPUs=0, queue=None, scratch='',
type=ANALYSIS,
    userSubroutine='', waitHours=0, waitMinutes=0)

#mdb.jobs['cubeJob'].submit(consistencyChecking=OFF)
#mdb.jobs['cubeJob'].waitForCompletion()
```

Figura 48: Script 3.3 [obtención propia].

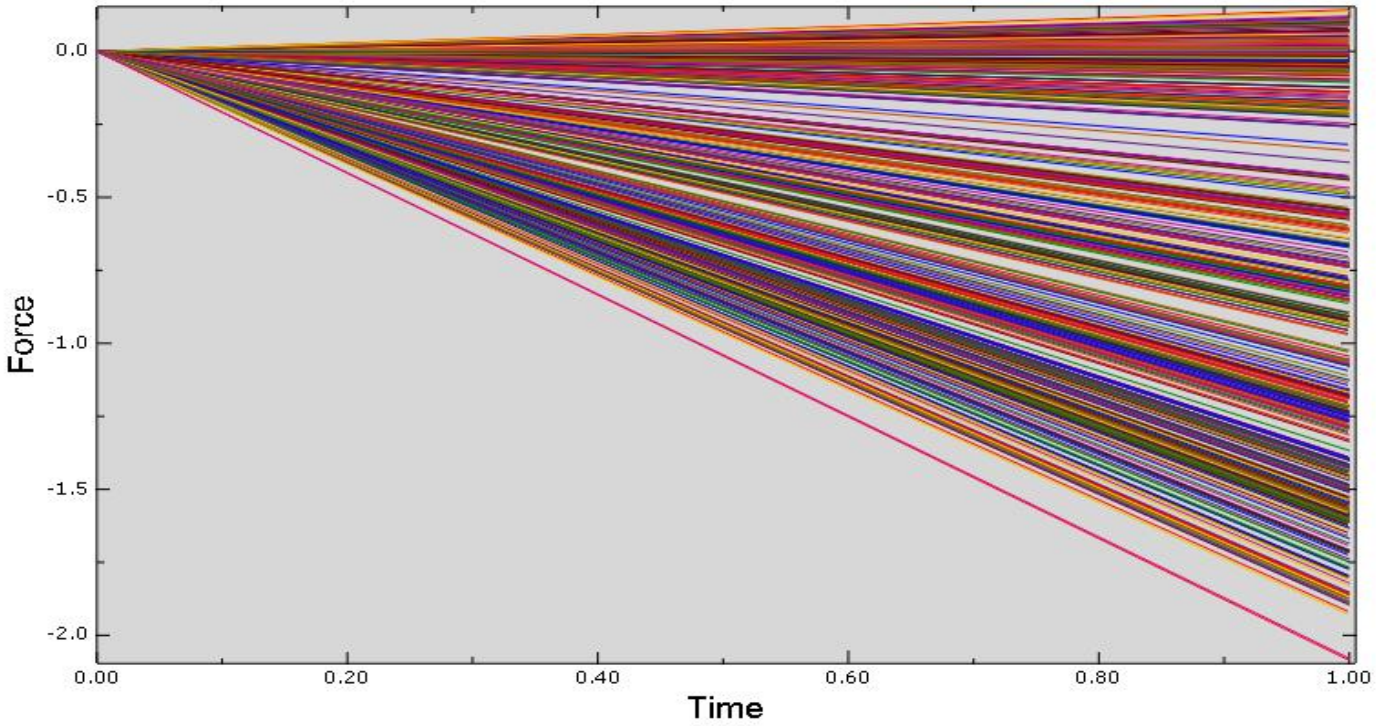
El método *job* define un objeto 'job' para el modelo. Es necesario entre otras cosas dar un nombre a este objeto y elegir el modelo que ha de analizar. También se pueden especificar parámetros como el uso de memoria que quiere consumir, o una pequeña descripción del análisis a realizar.

El comando *submit* inicia y completa la ejecución del análisis, realizándose de este modo los cálculos pertinentes. Ningún argumento ha de ser especificado, aunque si hay varios argumentos opcionales. En este caso simplemente hemos desactivado la comprobación de consistencia de la configuración del análisis ya que no era necesaría.

El último comando es waitForCompletion y es útil para hacer que la ejecución se halle bajo control hasta que finalice completamente.

Una vez ejecutado todo este código, la definición del modelo y su análisis han terminado. Ya solo falta comprobar y anotar los resultados que nos permitan realizar los cálculos necesarios una vez completado el 'job' y repetir todo el proceso hasta obtener todos los valores para cada uno de los 5 modelos.

CAPÍTULO V: ANÁLISIS FINAL. RESULTADOS



El objetivo de este capítulo es exponer los resultados observados tras realizar los distintos análisis en Abaqus y explicar la obtención de las constantes que se perseguía conseguir con este trabajo. También se incluye una valoración e interpretación general sobre esos resultados.

En primer lugar hablaremos sobre el análisis de malla llevado a cabo para decidir el tamaño del RVE y el número de elementos óptimo.

5.1 Análisis de malla

En programas como Abaqus y en problemas que se resuelvan por el método de elementos finitos en general, es necesario realizar un análisis de la malla para optimizarla y obtener unos resultados fiables. Como se explicó en un capítulo anterior, si los elementos usados en la discretización eran demasiado grandes, la interpolación de valores para puntos muy alejados de los nodos no era suficientemente precisa. Por otro lado al tratarse de un programa informático, se busca un número de nodos y elementos equilibrado, no demasiado grande para reducir en la medida de lo posible el coste computacional del análisis.

Para este análisis de malla se ha elegido un modelo simplificado para facilitar el mallado con elementos más grandes de lo esperado, que sirva de ejemplo mirando casos parecidos en la bibliografía, con un cubo base de un milímetro de lado. Se generaron para este modelo numerosas mallas con elementos que variaron entre $5*10^{-4}$ y $5*10^{-5}$ (valores límite con que el programa generaba satisfactoriamente la malla), dando lugar a modelos con un número de elementos que oscilaba entre valores de 7140 y 83722 elementos respectivamente. Cada uno de estos modelos fue analizado y se estudió la variación de los resultados hasta que estos se estabilizaban en torno a determinados valores, permitiéndonos elegir el modelo que mejor cumplía con las características que buscábamos.

Los resultados obtenidos fueron los siguientes:

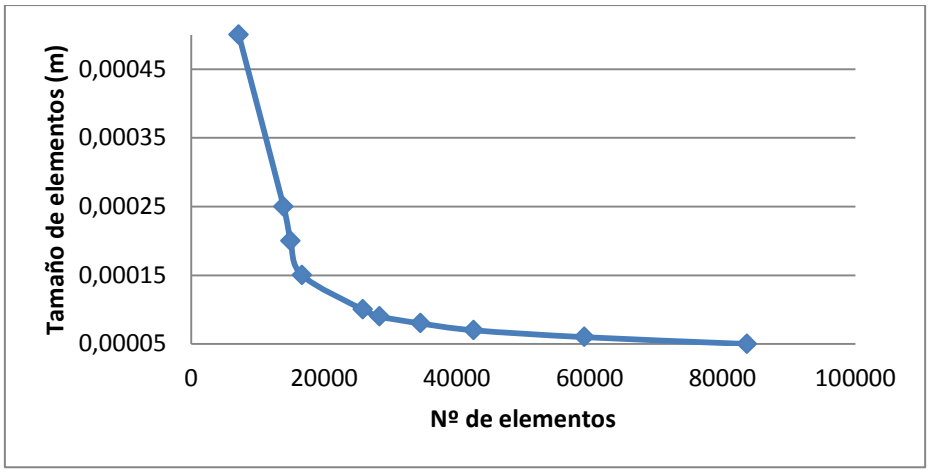


Figura 49: Gráfica nº elementos vs. Tamaño de elementos [obtención propia]

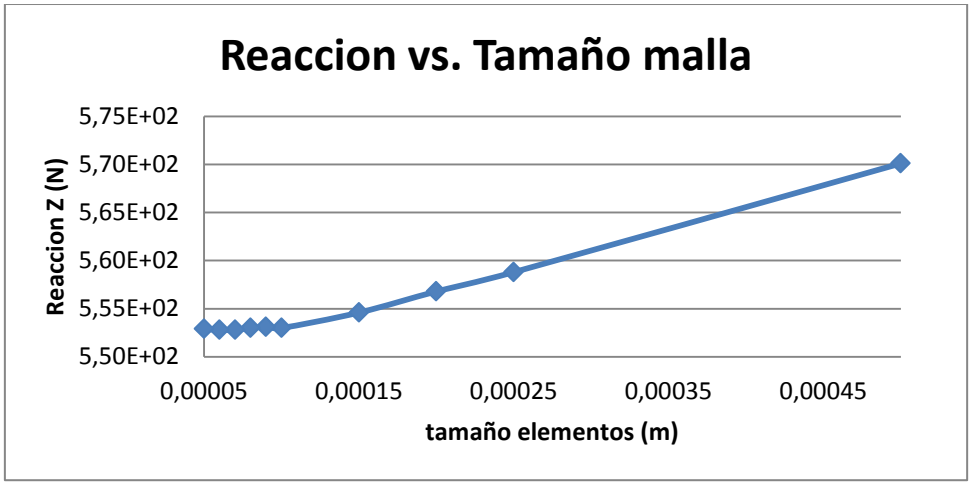


Figura 50: Reacción Z vs. Tamaño de elementos [obtención propia]

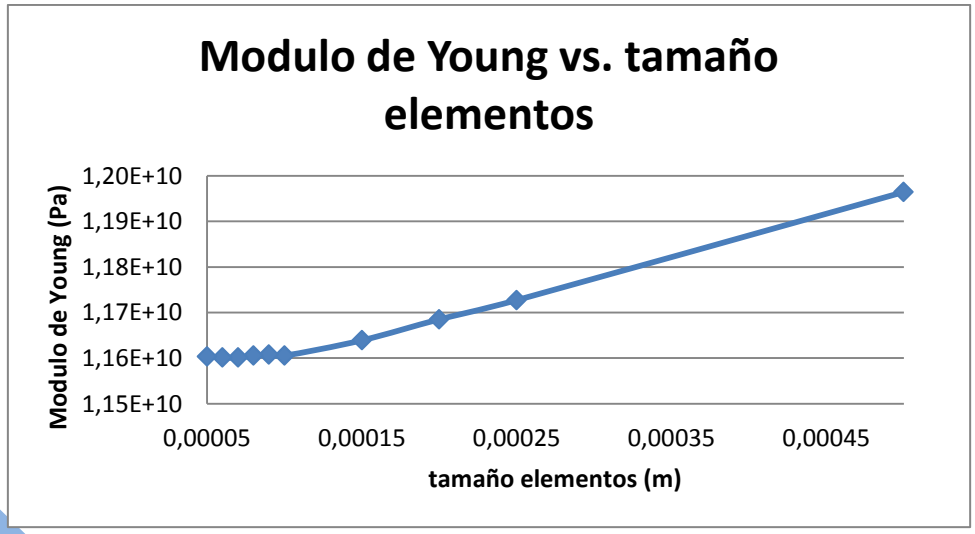


Figura 51: Módulo de young vs. Tamaño de elementos [obtención propia]

Se observa con estos resultados que tanto la reacción Z como el módulo de Young obtenidos se estabilizan a partir de un tamaño de elementos de $1e*10^{-4}$ y $9*10^{-5}$, a partir de los cuales el resultado apenas varía según disminuye el tamaño de los elementos. Para los modelos finales elegimos finalmente el tamaño de $9*10^{-5}$.

5.2 Análisis del RVE

Una vez realizado el análisis de malla, se procedió a realizar el análisis del tamaño del volumen representativo para encontrar el tamaño óptimo del modelo. Se buscaba con este análisis un modelo no demasiado pequeño, ya que produciría fracción del volumen ocupado por las osteonas no sería correcta, (y por lo tanto el volumen no representaría fielmente la realidad) pero tampoco demasiado grandes, puesto que una vez se estabilice la fracción volumétrica de osteonas y con ello las propiedades del modelo, cuanto mayor tamaño tenga el modelo más elementos albergará y mayor coste computacional supondrá.

Para este análisis se construyeron modelos con valores del tamaño del lado entre $1*10^{-4}$ y $1{,}25*10^{-3}$, los cuales dieron lugar a modelos que contenían desde una a 25 osteonas.

Al igual que en el análisis anterior se analizaron todos los modelos creados observando los resultados y buscando el punto en que se estabilizasen para elegir el más pequeño de los modelos que reportara un valor estabilizado.

Los resultados obtenidos fueron los siguientes:

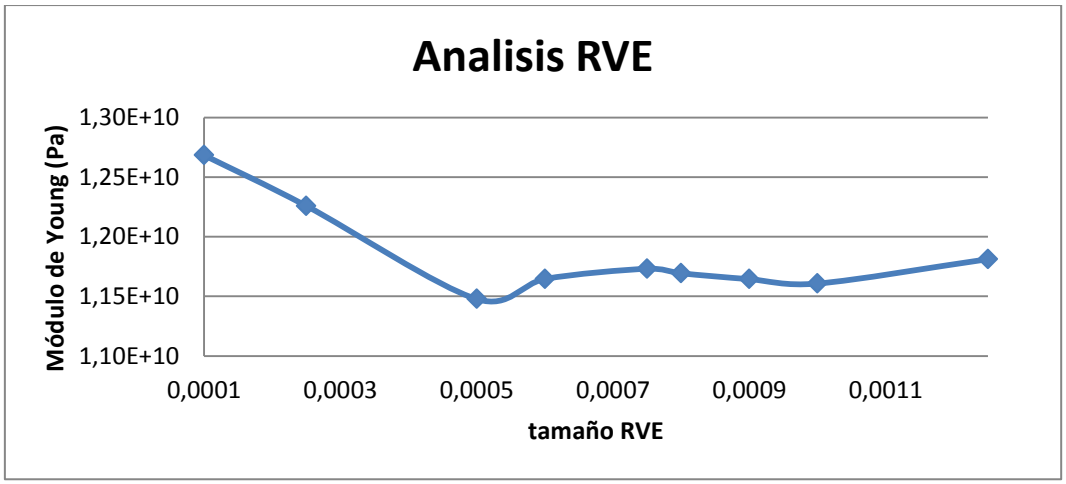
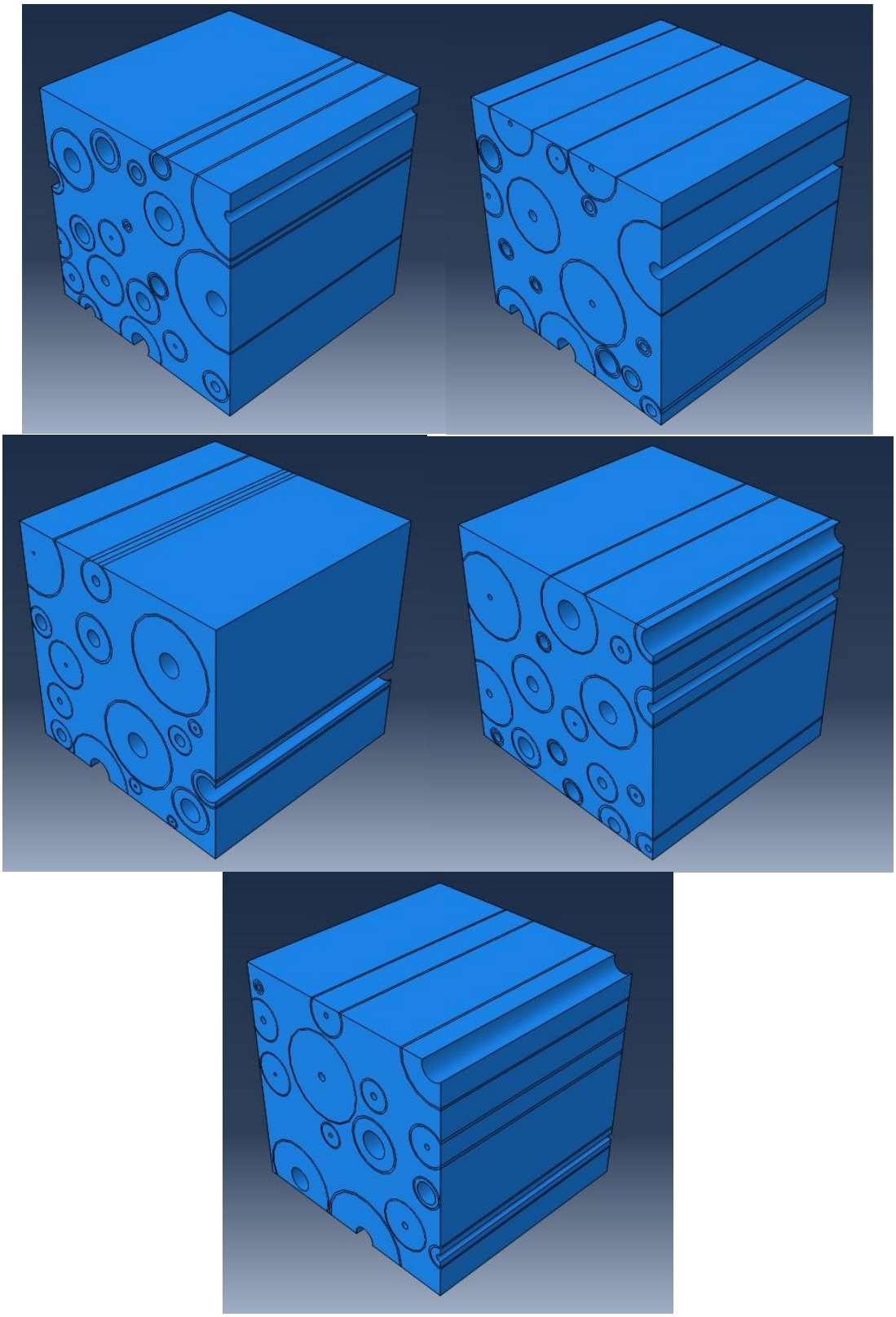


Figura 52: Módulo de young vs. Tamaño RVE [obtención propia]

Se observa en los resultados como al principio el modelo es demasiado pequeño, con una fracción de osteonas muy por debajo de la correcta. Según aumenta el tamaño los valores del módulo de Young se estabilizan en torno a 1,17-1,16 GPa. Finalmente para el modelo se decidió utilizar un tamaño de lado de 0,001m, con un valor comprendido dentro de los valores aceptables, y un número de osteonas suficiente como para poder ajustar la fracción de estas al 65 por ciento.

5.3 Modelos Finales

Una vez conocidos los resultados de los análisis de malla, se pudo proceder y crear los 5 modelos definitivos sobre los que realizar los distintos análisis y comparar sus resultados. Cada modelo final fue sometido a 9 análisis: 3 a tracción, uno en cada dirección principal, y 6 a cortadura implementando todas las combinaciones de cortadura posibles, haciendo un total de 45 análisis distintos con los que contrastar las propiedades obtenidas. Tras una serie de intentos estos son los 5 modelos definitivos:



Figuras 53-57: Imágenes de los 5 modelos finales conseguidos tras los análisis de malla y RVE [obtención propia].

5.4 Resultados

Como se señaló en capítulos anteriores, las variables de las cuales se quiere obtener datos proporcionados por Abaqus son básicamente desplazamientos y reacciones, a partir de los cuales se realizarán una serie de cálculos para obtener las constantes elásticas. Estos cálculos se explican en el capítulo siguiente.

A continuación se muestras y evalúan los distintos resultados obtenidos para los 5 distintos modelos:

5.4.1 Reacciones

Para obtener estas reacciones el proceso a seguir era siempre similar: Se obtenía una gráfica a partir de todos los nodos de la cara empotrada correspondiente, con los valores de la reacción en la dirección correspondiente para a continuación sumar todos ellos y obtener la fuerza total de la reacción.

El aspecto deformado de un modelo sometido a tracción pura es el siguiente:

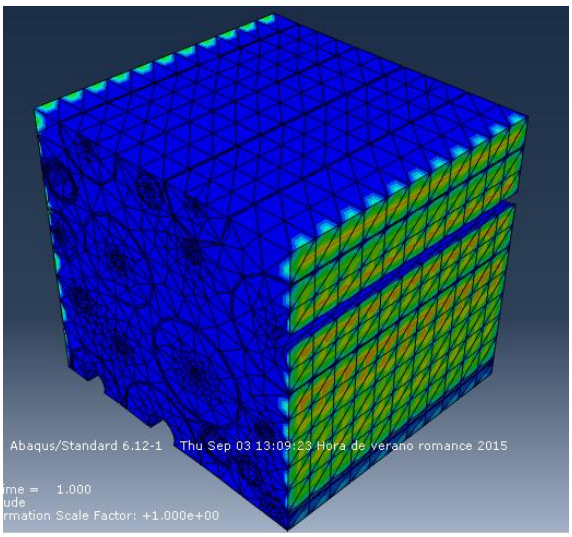
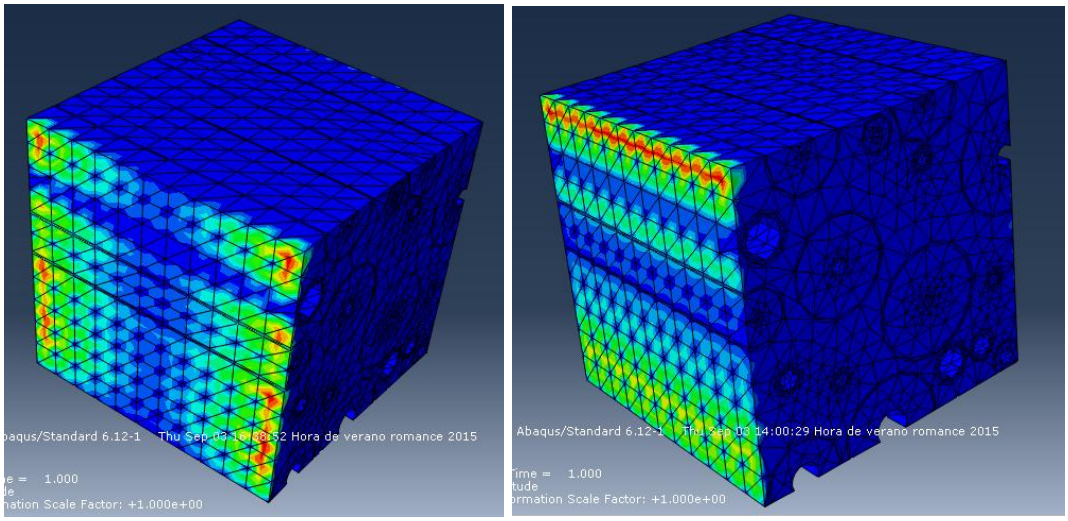


Figura 58: Aspecto deformado del modelo en un caso de tracción [obtención propia].

Se puede observar que tan solo dos de las caras están sometidas a un esfuerzo importante comparado con las demás: la cara del empotramiento y la cara donde se impone el desplazamiento. Además comparado con el modelo sin cargas se produce un alargamiento en la dirección de tracción y un acortamiento en las dos direcciones transversales.

El aspecto visual de un modelo sometido a cortadura es el siguiente:



Figuras 59-60: Aspecto del modelo sometido a casos de cortadura [obtención propia].

Se observa en la cara empotrada un pequeño patrón en la propia dirección de cortadura, en el que los valores son máximos en los extremos y hay un punto con valor en el centro de la cara, correspondiendo este patrón con la teoría de mecánica de materiales. Los modelos además se encuentran ligeramente deformados en la dirección en que se aplica la carga de cortadura.

Las gráficas a continuación muestran los valores obtenidos para los tres casos de tracción pura en cada uno de los 5 modelos:

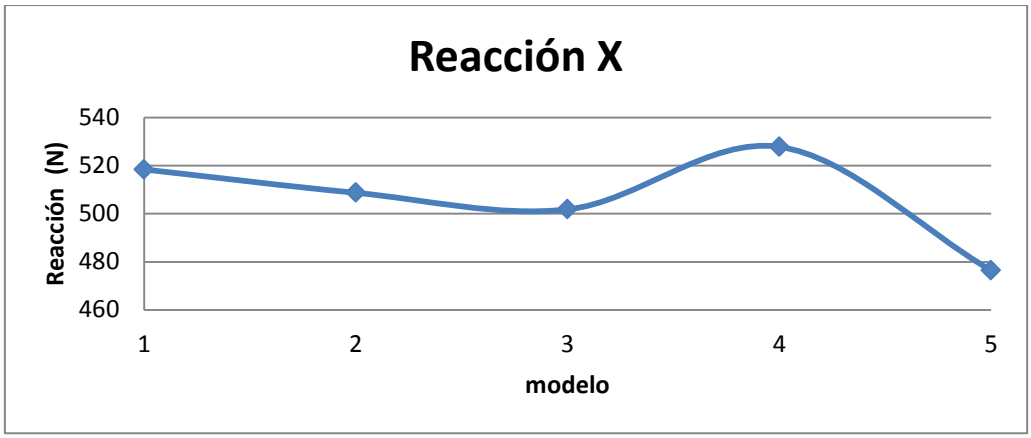


Figura 61: Reacción X según modelo [obtención propia].

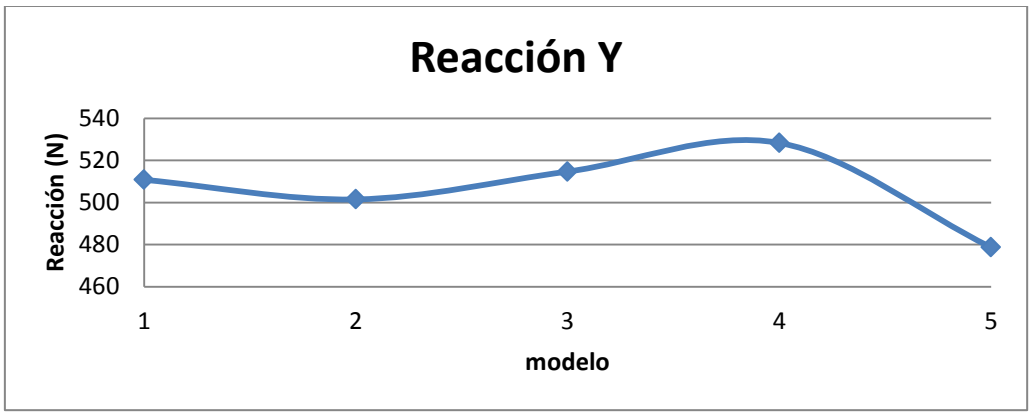


Figura 62: Reacción Y según modelo [obtención propia]

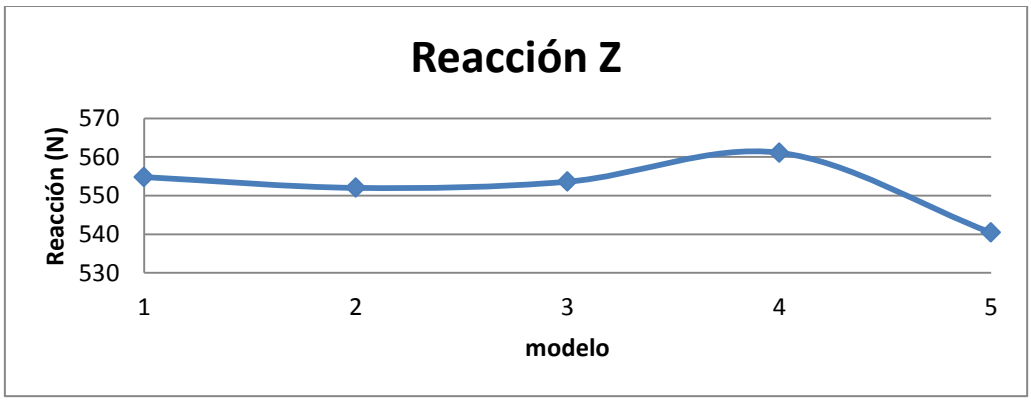


Figura 63: Reacción Z según modelo [obtención propia]

Se observa que las reacciones no presentan una gran diferencia entre valores mínimos y máximos. La diferencia entre valores se ve influenciada por factores

como la disposición de las osteonas, o la diferencia entre los valores de las distintas áreas. Se observa también que son las reacciones en dirección Z (dirección paralela a las osteonas) las que tienen un mayor valor, lógico, puesto que es en esta dirección aquella en la que se espera una mayor rigidez.

El mayor valor observado toma un valor de unos 561,1 Newtons y se produce en uno de los casos de tracción en dirección Z. El mínimo valor corresponde a uno de los estados de carga en tracción X y es de 476,4 Newtons.

Se incluyen a continuación las gráficas correspondientes a los 6 distintos casos de esfuerzos de cortadura, en total 6 casos por modelo:

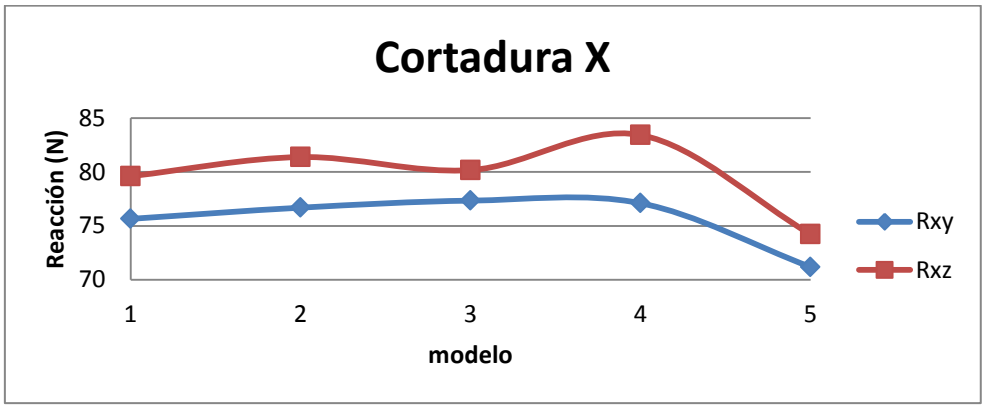


Figura 64: Reacciones frente a cortadura según modelo [obtención propia]

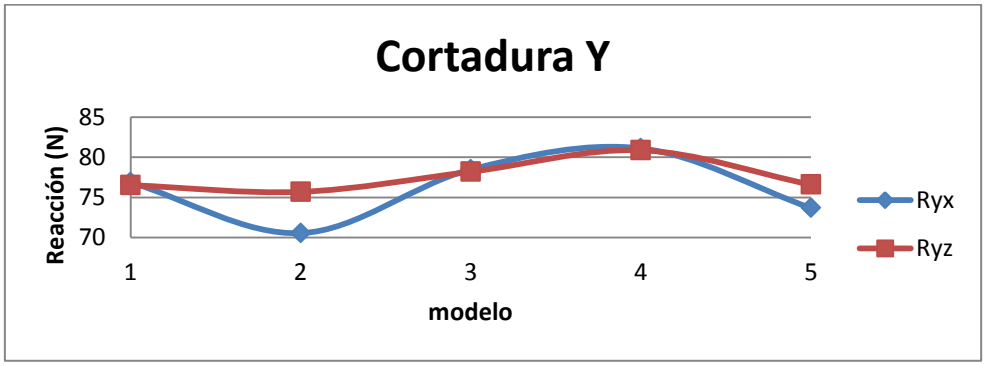


Figura 65: Reacciones frente a cortadura según modelo [obtención propia]

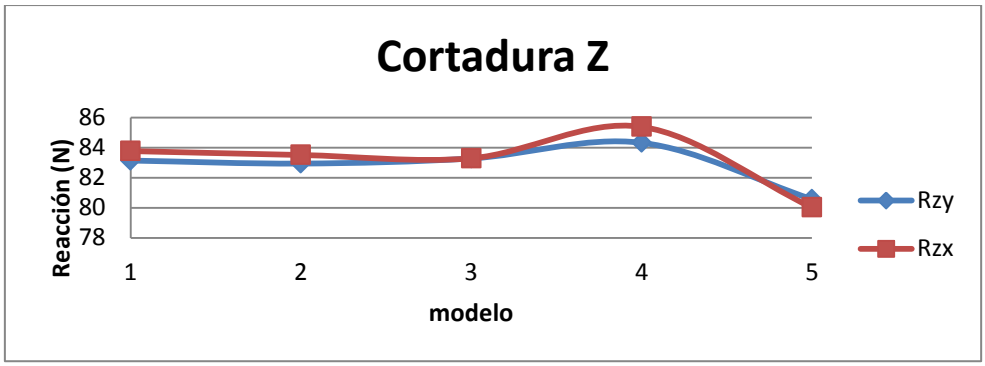


Figura 66: Reacciones frente a cortadura según modelo [obtención propia]

Para los valores de las reacciones en los casos de cortadura tampoco se observa una gran variación entre valores máximos y mínimos. Al igual que para casos de tracción también son las reacciones relacionadas con la dirección longitudinal de las osteonas las que adquieren valores mayores. El máximo valor se da para el caso de cortadura ZY y toma un valor de 84,32 Newtons. El menor se da en un caso de cortadura XY y su valor es de 71,19 Newtons. Se puede remarcar como todos estos valores son de un orden menor que los obtenidos a tracción.

Como las osteonas suponen en parte realizar un vaciado en la pieza, también era necesario obtener los datos sobre el área de cada cara empotrada, para poder calcular la tensión. Estos resultados se muestran en la siguiente tabla, y cada uno corresponde al área de la cara empotrada obtenida en los casos de tracción pura:

Tabla con los valores de las áreas de las caras empotradas. Todos los valores de 1e-6 corresponden con caras lisas completamente, no afectadas por ningún canal de Havers [obtención propia].

Modelo	Area X (m^2)	Area Y (m^2)	Area Z (m^2)
1	1,00E-06	8,30E-07	9,48E-07
2	1,00E-06	8,87E-07	9,48E-07
3	1,00E-06	1,00E-06	9,45E-07
4	1,00E-06	9,05E-07	9,60E-07
5	9,42E-07	8,50E-07	9,25E-07

5.4.2 Desplazamientos

En caso de los desplazamientos, se han evaluado los nodos de la cara opuesta a la empotrada, una cara libre capaz de deformarse y desplazarse. Para obtener estos valores, se seleccionaban todos los nodos de la cara correspondiente y se graficaban los valores de desplazamiento en la dirección correspondiente, para luego obtener un valor medio para todos los nodos con el que proceder en los cálculos.

Los desplazamientos obtenidos son los transversales en las dos direcciones restantes transversales en cada caso de tracción pura, para poder obtener los coeficientes de Poisson. Las siguientes gráficas representan los valores obtenidos:

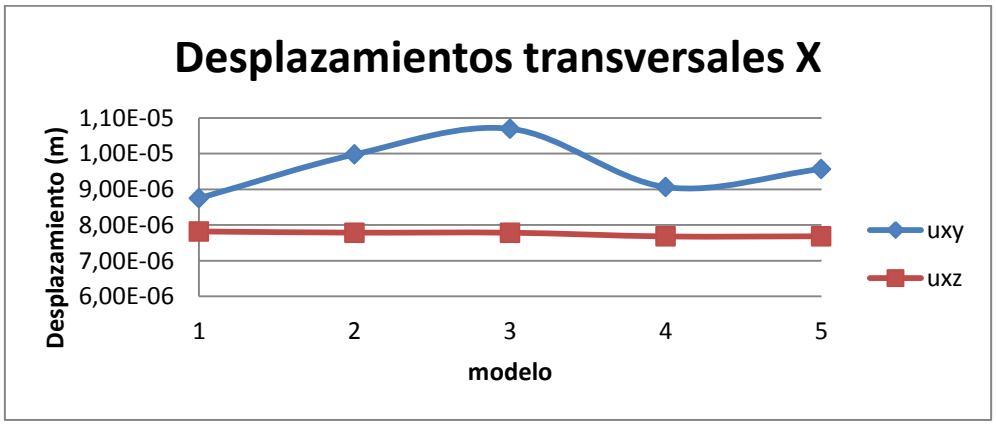


Figura 67: Desplazamientos transversales a X según modelo [obtención propia]

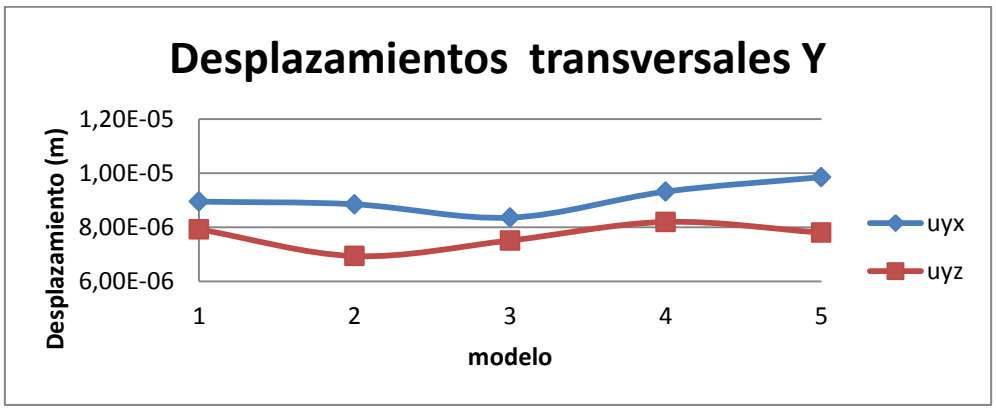


Figura 68: Desplazamientos transversales a Y según modelo [obtención propia]

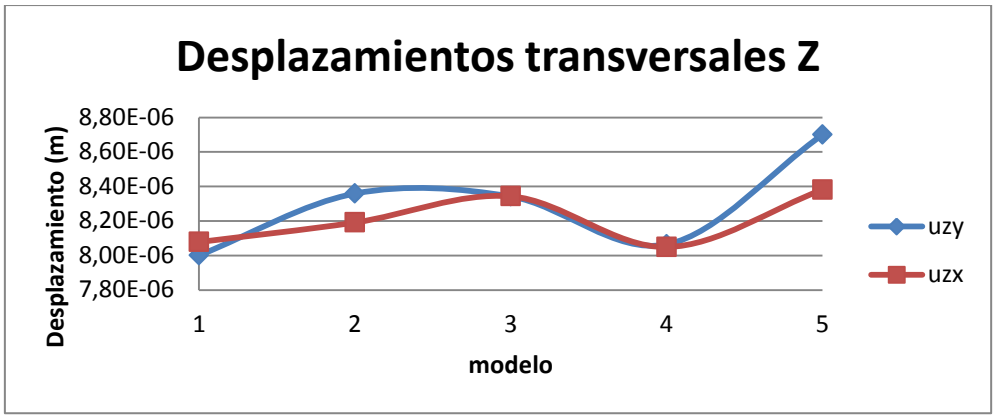


Figura 69: Desplazamientos transversales a Z según modelo [obtención propia]

Es evidente, tal como se esperaba que todos los valores de estos desplazamientos están muy por debajo del valor de 0,05 impuesto en la dirección longitudinal correspondiente. Como ya sabemos, la mayor rigidez es la que se espera en la dirección Z, por lo que valores en dirección XY son los que se espera tengan mayor valor en el caso de los desplazamientos. En efecto, el mayor valor se da para el caso de tracción X en la dirección Y, y toma un valor de $1,07*10^{-5}$ metros, mientras que el menor desplazamiento transversal es de $6,94*10^{-6}$ y se da para el caso de tracción en dirección Y en dirección transversal Z. Nótese que todos los valores de la tabla son valores de contracción, es decir de valor negativo, en concordancia con la definición del efecto Poisson.

5.5 Cálculo de las constantes elásticas

Una vez obtenidos los resultados de desplazamientos y reacciones a partir de Abaqus, se pueden realizar los cálculos pertinentes para obtener las distintas constantes elásticas, objetivo del trabajo.

5.5.1 Módulos de Young

Para obtener el valor de los módulos de Young se ha utilizado la Ley de Hooke, ley básica de la elasticidad de materiales: $\varepsilon = \frac{\sigma}{E}$ donde **ε** es la deformación elástica, **E** el módulo de Elasticidad y σ la tensión obtenida de la siguiente forma: $\sigma = \frac{F}{A}$ donde **F** es la fuerza y **A** el área sobre el que esta actúa.En este caso A es el área de la cara empotrada, F la reacción correspondiente, y ε el desplazamiento impuesto, con un valor de 0,05. Una vez conocidos todos estos valores se puede operar y despejar el valor del módulo de young

Los valores finales obtenidos para los modulos de Young se incluyen en las siguientes gráficas que ilustran además la variabilidad de los resultados:

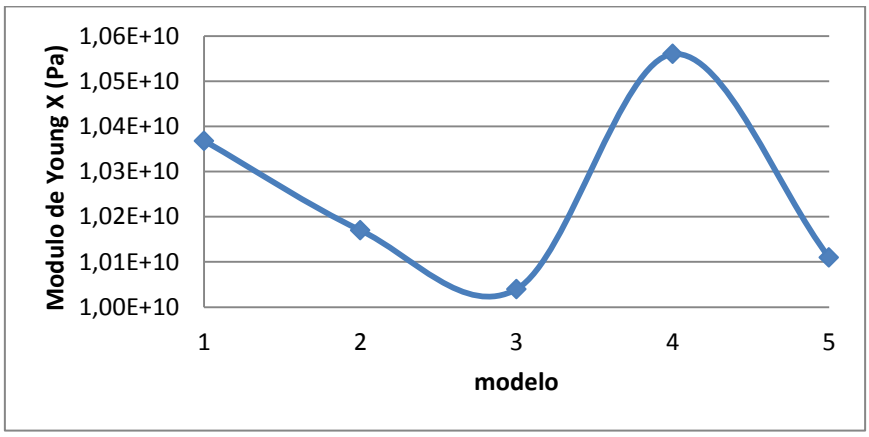


Figura 70: Módulo de Young en dirección X según modelo [obtención propia].

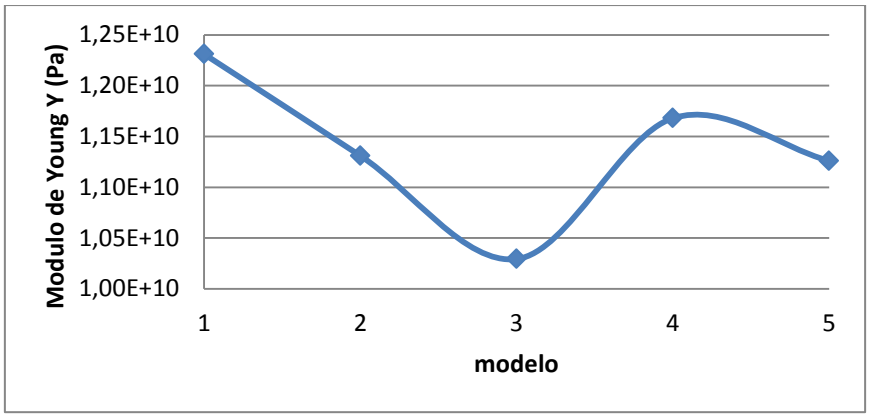


Figura 71: Módulo de Young en dirección Y según modelo [obtención propia].

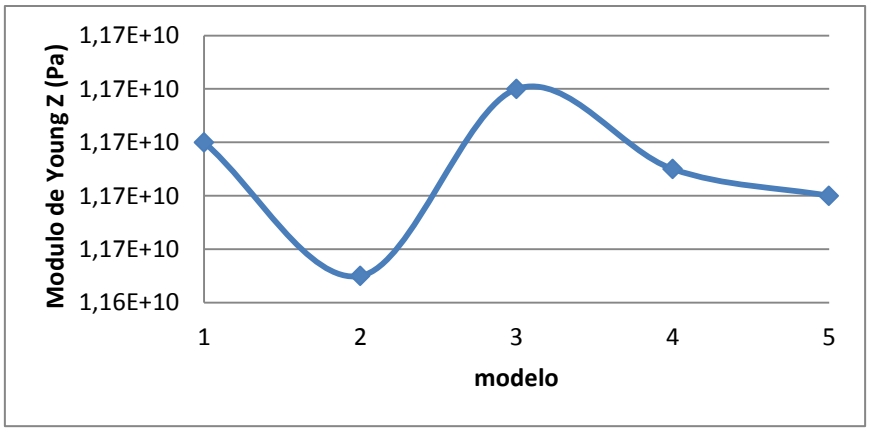


Figura 72: Módulo de Young en dirección Z según modelo [obtención propia].

Se observa que los valores obtenidos para el módulo de Young toman valores intermedios entre los definidos para las osteonas, líneas cementantes y matriz intersticial. Como en el resto de resultados anteriores, y como es lógico, son los valores en dirección Z los que reportan una mayor rigidez. También son estos valores los que presentan una menor variabilidad, tomando todos unos valores de unos 11,7 GPA. Se observa también que los valores obtenidos en dirección Y son también ligeramente mayores que los obtenidos en dirección X

5.5.2 Coeficientes de Poisson

Para cuantificar los valores para cada uno de los coeficientes de Poisson, una vez conocidos todos los desplazamientos, se ha divido simplemente la deformación transversal entre la longitudinal. La longitudinal en este caso es la impuesta como condición de valor 0,05. La transversal se obtiene a partir de los desplazamientos transversales a partir de la siguiente fórmula: $\varepsilon = \frac{\Delta l}{L}$. Los resultados de los coeficientes de Poisson obtenidos son los siguientes:

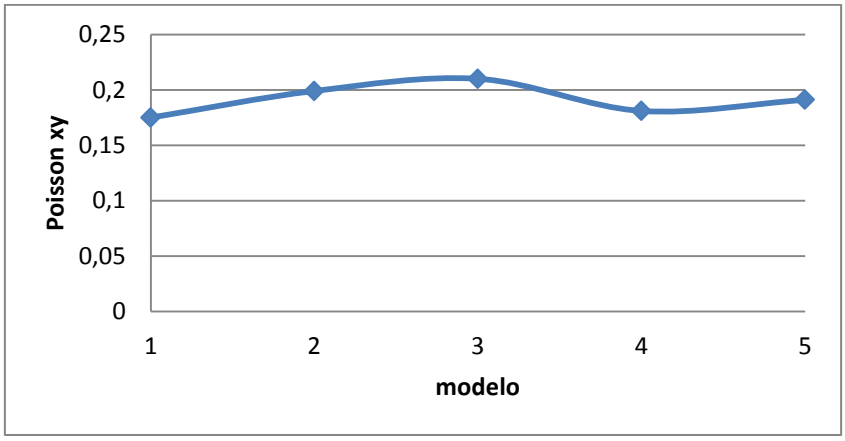


Figura 73: Coeficiente de Poisson en dirección XY según modelo [obtención propia].

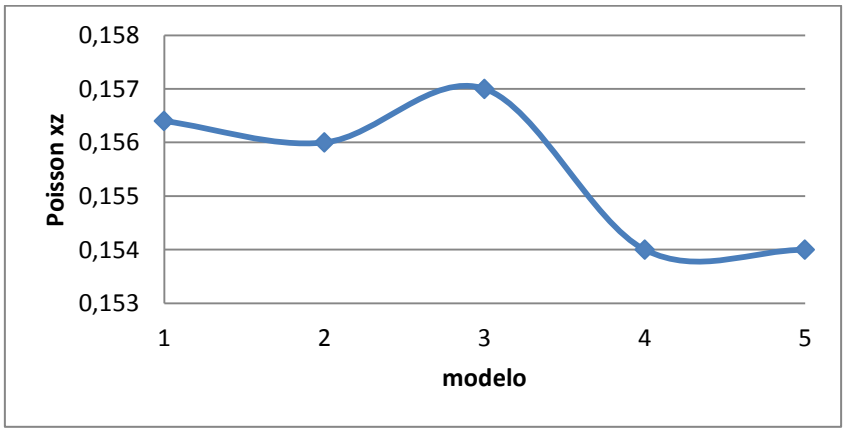


Figura 74: Coeficiente de Poisson en dirección XZ según modelo [obtención propia].

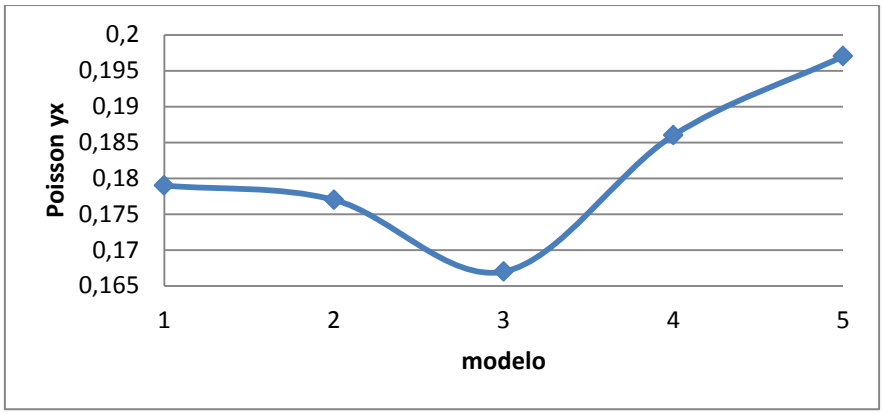


Figura 75: Coeficiente de Poisson en dirección YX según modelo [obtención propia].

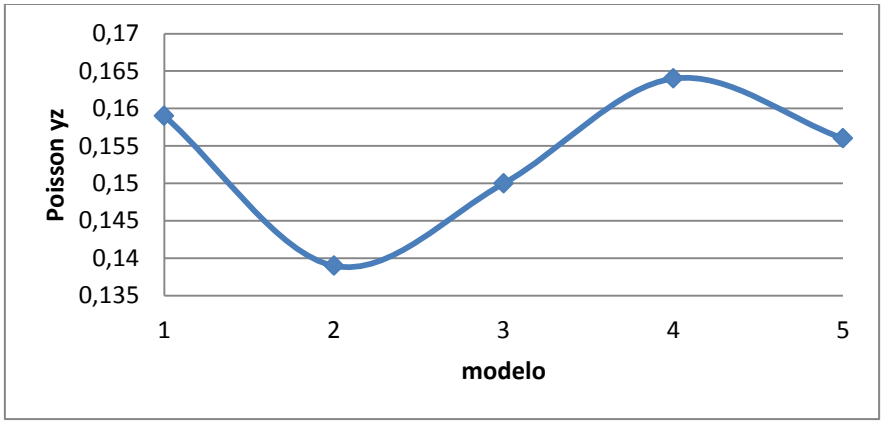


Figura 76: Coeficiente de Poisson en dirección YX según modelo [obtención propia].

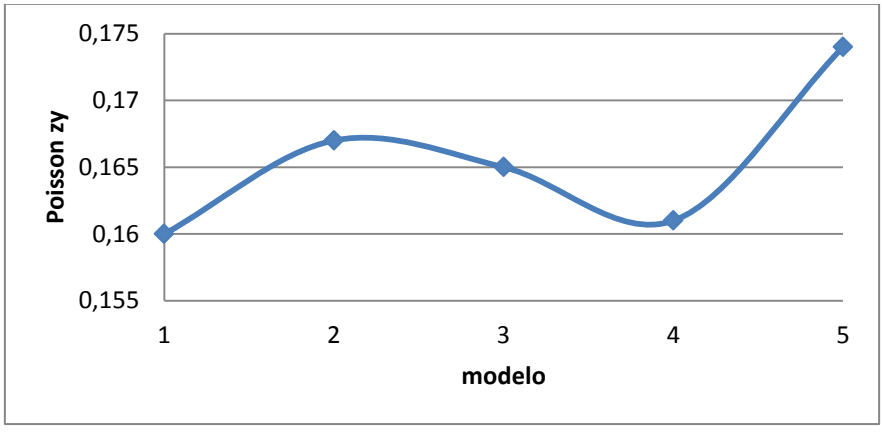


Figura 77: Coeficiente de Poisson en dirección ZY según modelo [obtención propia].

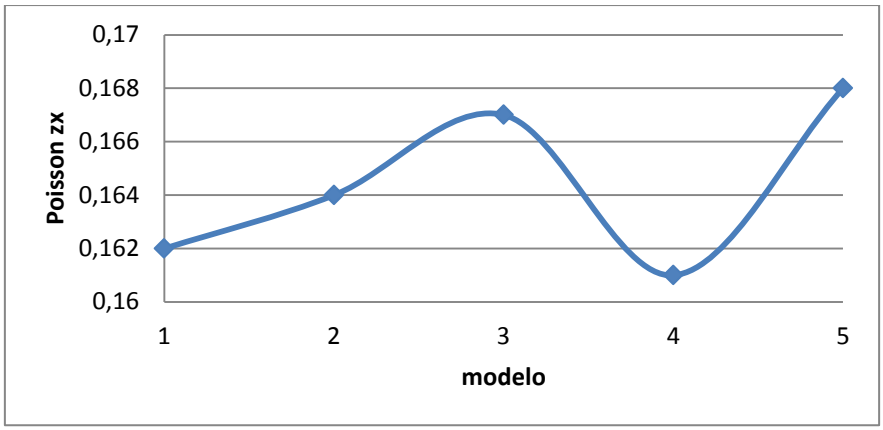


Figura 78: Coeficiente de Poisson en dirección ZX según modelo [obtención propia].

De acuerdo con los valores obtenidos para los desplazamientos transversales, los valores del coeficiente de Poisson de mayor valor son los obtenidos en dirección XY, ya que en esta dirección para un mismo desplazamiento longitudinal se producen mayores desplazamientos transversales. El mayor valor del coeficiente de Poisson observado vale 0,21 y se da en un caso de tracción X en dirección Y. El menor valor es de 0,139 para un caso de tracción en Y en dirección transversal Z. Salvo en un caso, los valores del coeficiente de Poisson no sobrepasan un valor de 0,2, por lo que se podría afirmar que la deformación transversal no supondrá más de un 20% de la longitudinal.

5.5.3 Módulos de cortadura:

Los módulos de cortadura se obtienen con una fórmula análoga a la del módulo de Young: $\gamma = \frac{\tau}{G}$, donde **γ** es la deformación angular, **τ** es el esfuerzo de cortadura y **G** es el módulo de cortadura. Para obtener el valor de los esfuerzos de cortadura, conocidas las reacciones correspondientes, se dividían entre el área correspondiente. La deformación angular se obtiene calculando el ángulo que se forma al deformarse el modelo en la dirección de cortadura (en la que se impone el desplazamiento de 0,05). Si por ejemplo hablamos del caso de cortadura XY, la cortadura se impone en dirección Y, por lo que el ángulo a calcular es el que forma el plano XZ deformado con el no deformado,

perpendicular a los ejes de coordenadas. Se produce un pequeño alargamiento en dirección X, tenido en cuenta a la hora de calcular el ángulo.

Los valores obtenidos para los módulos de cortadura son:

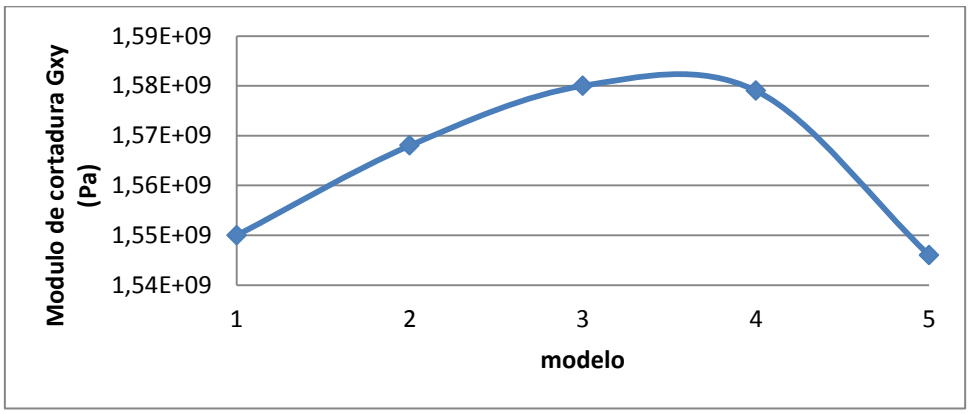


Figura 79: Módulo de cortadura en dirección XY según modelo [obtención propia].

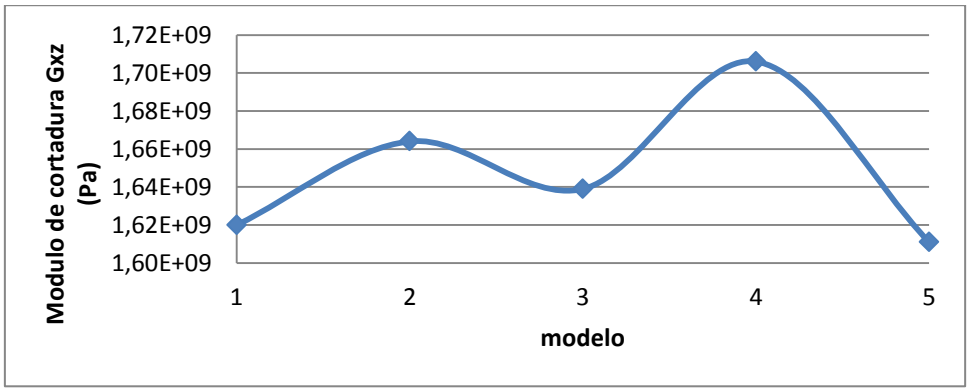


Figura 80: Módulo de cortadura en dirección XZ según modelo [obtención propia].

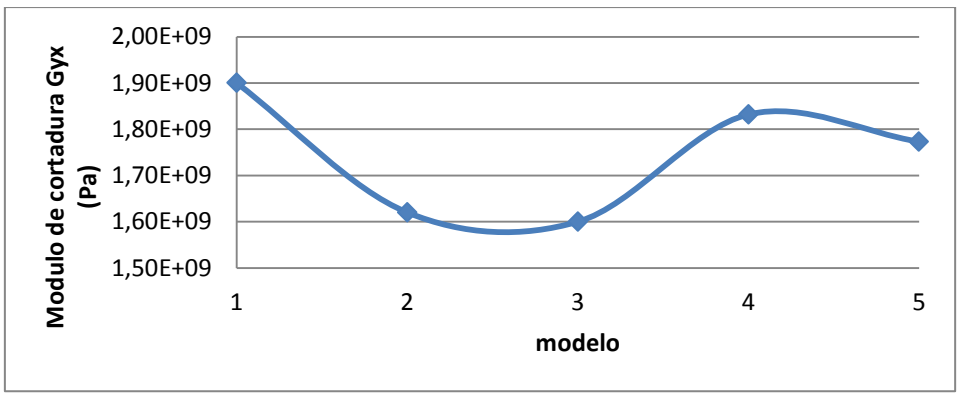


Figura 81: Módulo de cortadura en dirección YX según modelo [obtención propia].

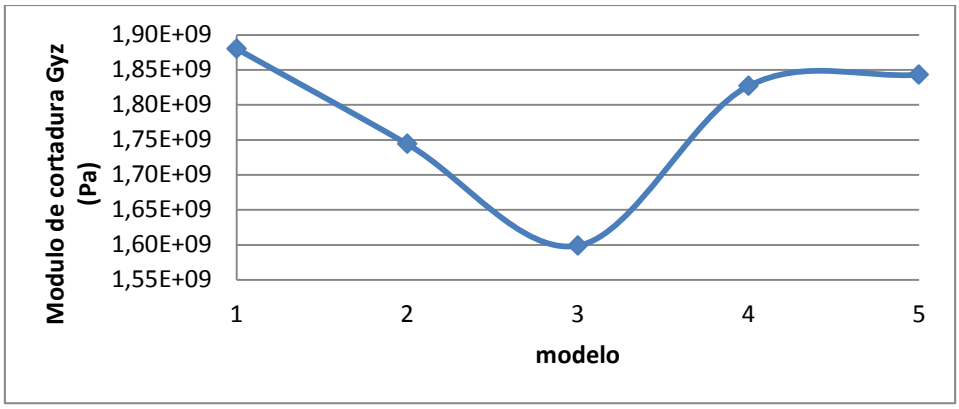


Figura 82: Módulo de cortadura en dirección YZ según modelo [obtención propia].

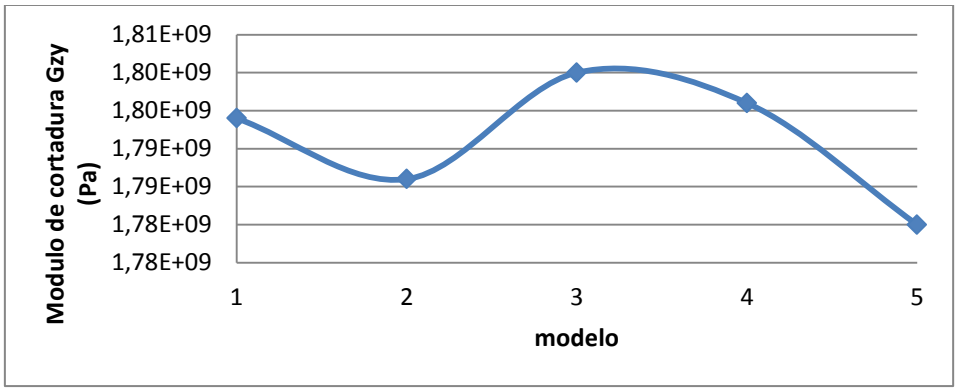


Figura 83: Módulo de cortadura en dirección ZY según modelo [obtención propia].

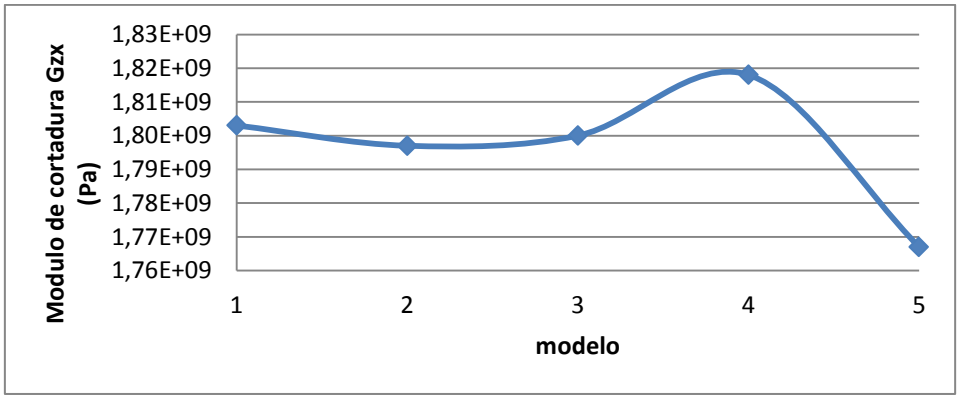
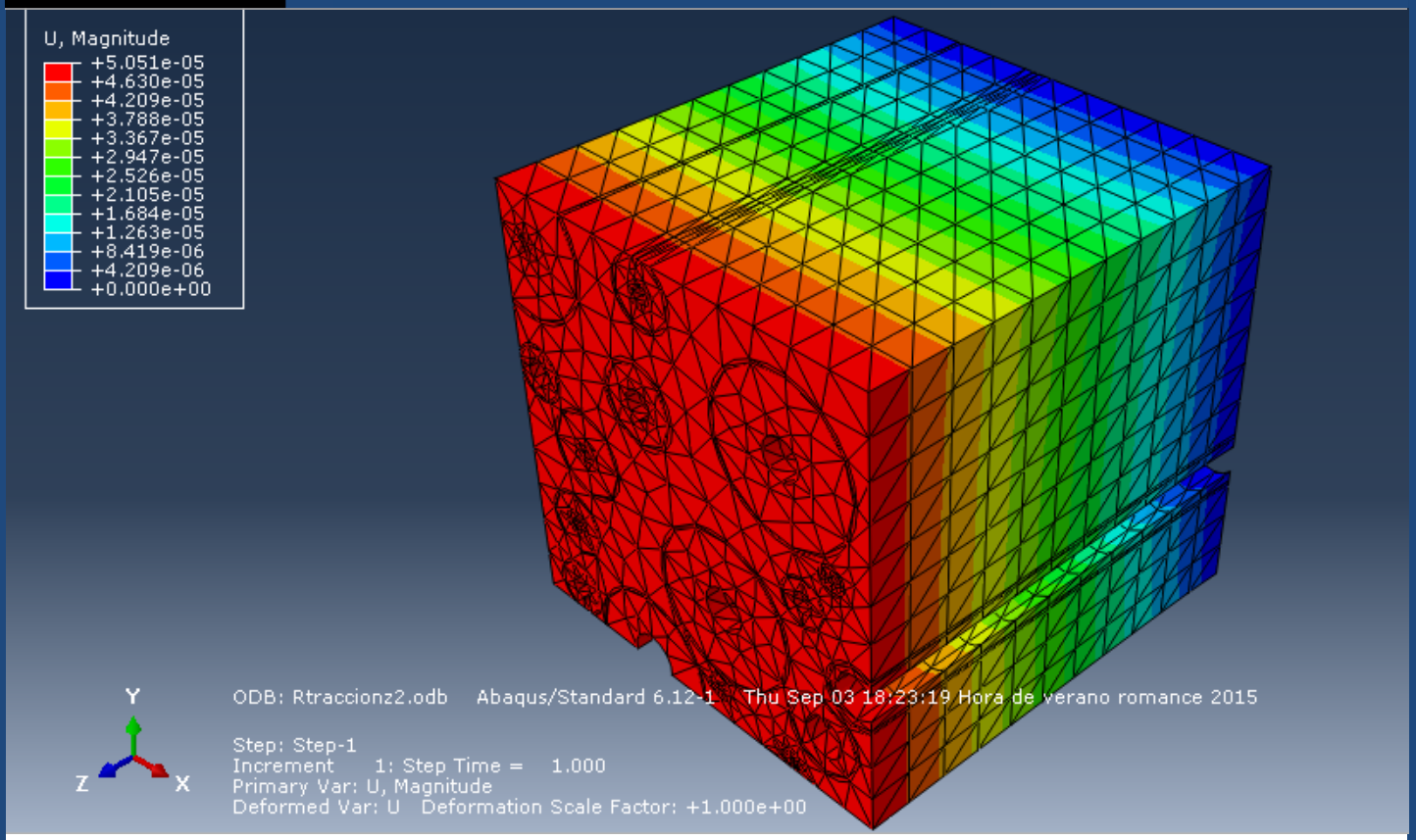


Figura 84: Módulo de cortadura en dirección ZX según modelo [obtención propia].

Por último para los módulos de cortadura se observan valores un orden menor que los correspondientes módulos de elasticidad. El menor valor se da en el

caso XY, y su valor es de 1,55 GPA. El mayor por su parte es de 1,9 GPa y se da en el caso YX, aunque por lo general son los valores relacionados con el eje Z los de un mayor valor.

CAPITULO VI: CONCLUSIONES



6.1 Conclusiones

Una vez obtenidos y evaluados los resultados las conclusiones que se pueden extraer son las siguientes:

- Se ha logrado desarrollar un modelo numérico mediante el uso del software Abaqus CAE, y apoyándonos en el uso de scripts programados en Python para automatizar en la medida de lo posible el generación de estos modelos con distribuciones aleatorias de osteonas. Automatizando este proceso, hemos conseguido crear fácilmente modelos representativos de volúmenes representativos de tejido óseo cortical con dimensiones y propiedades reales y estudiar así el comportamiento mecánico de este tipo de tejido.

-Se ha realizado un análisis de la sensibilidad de la malla, muy importante en problemas que se resuelvan por el método de elementos finitos, obteniendo el tamaño óptimo para este caso, que proporcionase unos resultados suficientemente precisos con un coste computacional lo más reducido posible.

-De forma similar se ha estudiado la influencia del tamaño del RVE y este análisis numérico, no siendo fiel a la realidad para tamaños muy pequeños en que el número de osteonas es muy reducido. Para volúmenes mayores la fracción de osteonas se estabilizaba, por lo que para reducir el coste computacional del análisis se eligió el volumen más pequeño que representase correctamente el tejido cortical.

-Se ha alcanzado también uno de los principales objetivos, el de obtener las constantes elásticas en todas las direcciones del modelo que permitirán modelizar este tipo de tejido como un material uniforme en trabajos y proyectos posteriores. Las propiedades obtenidas son al fin y al cabo una combinación en la cual toman parte en mayor o menor medida los 3 tipos de materiales que constituyen el modelo.

-En cuanto a las mayores tensiones obtenidas siempre se localizan en las caras empotradas, como es lógico, pero más concretamente en los nodos del perímetro exterior de las líneas cementantes, que presentan el módulo de elasticidad más pequeño de todo el modelo por lo que para un mismo desplazamiento la tensión será mayor. Por otra parte las deformaciones de

mayor valor se encuentran en la cara libre opuesta a la empotrada, es decir, la cara donde se imponen la condición del desplazamiento. Más concretamente suelen darse los mayores valores en zonas próximas a los vértices o aristas de la cara que se encuentra libre.

6.2 Futuros trabajos

Llegados a los puntos finales de este trabajo ya solo queda hablar de futuros proyectos que pueden ser realizados apoyándose o basándose en este mismo. Algunos posibles trabajos que realizar a continuación de este son:

-Evidentemente, uno de los primeros proyectos que pueden derivar de este proyecto está directamente relacionado con los objetivos de este mismo: Utilizar los resultados obtenidos para simplificar la modelización de este tipo de tejido e implementar un modelo de hueso a escala real con dimensiones reales y las propiedades proporcionadas por este trabajo.

-Otro trabajo que podría ser sumamente útil sería el de analizar el comportamiento en régimen plástico de este tipo de tejido y obtener así la curva tensión-deformación completa para el tejido cortical apoyándose en ambos trabajos.

-Se puede estudiar también la inclusión de algún modo de daño o carga dinámica para analizar la respuesta del tejido cortical frente a estas situaciones: donde se produciría y concentraría el mayor daño, si se producen grietas donde se iniciarían, como se propagarían, etc.

-También podría resultar interesante someter el modelo a algún tipo de ciclo repetitivo continuado y estudiar su comportamiento a fatiga, ya que este tipo de situaciones son los que afrontan huesos como los de la pierna al andar o la espalda en el día a día de un ser humano.

-Otro proyecto relacionado podría ser el de estudiar la relación que existe entre el aumento de la porosidad en el tejido óseo y el empeoramiento de las propiedades mecánicas de este. Se podría realizar un estudio a fondo del fenómeno de la osteoporosis, que se produce por perdida de tejido cuando la

creación de tejido ósea no logra compensar la destrucción de este, y estimar los daños que esta puede ocasionar al debilitar el hueso hasta el punto de que una pequeña caída pueda dar lugar a una fractura [20].

-También es el grado de mineralización del hueso un factor determinante en las propiedades mecánicas del tejido óseo, volviendo más resistente pero más quebradizo al aumentar el contenido mineral. Un análisis a fondo de esta dependencia también podría resultar útil.

CAPITULO VII: PLANIFICACIÓN Y PRESUPUESTO

Como en cualquier proyecto real del mundo de la investigación, se ha optado por incluir en el proyecto una parte que en ciertos proyectos puede ser determinante para su realización: la planificación y el presupuesto. Conocer o al menos poder estimar la duración de las distintas tareas a realizar adquiere una gran importancia ya que nos permite organizar el tiempo del que disponemos de manera óptima, establecer fechas y estimar deadlines para nuestro proyecto. El presupuesto por su lado puede ser el punto decisivo de un proyecto, y nos permite valorar costes, y con ello evaluar si el proyecto sería rentable de algún modo.

7.1 Planificación

Se incluye a continuación una tabla resumen con las principales tareas llevadas a cabo y una estimación del tiempo que ocuparon. En cuanto a la tarea de escribir la memoria se fue haciendo poco a poco desde el momento en que se eligió el trabajo hasta el final de este.

Tabla con la planificación general del proyecto [obtención propia]:

actividad	periodo	Días dedicados
Búsqueda y elección de un tfg	noviembre-diciembre 2014	20-30 días
Obtención de información básica y artículos	febrero 2015	15 días
Primeros pasos en Python y scripting	marzo 2015	15 días
Creación de los scripts	abril-mayo 2015	25 días
Análisis de malla y RVE	junio 2015	7 días
Creación de los modelos definitivos	agosto 2015	2 días
Análisis de los 45 casos totales	agosto-septiembre 2015	7 días
Obtención de resultados	septiembre 2015	3 días
Terminar memoria del tfg	septiembre 2015	15 días

Sumando el total de todas las tareas se observa que la duración del proyecto se ha alargado en total alrededor 7 meses, sin tener en cuenta la búsqueda y elección de este.

7.2 Presupuesto

Para elaborar el presupuesto se incluirán las horas de trabajo totales invertidas tanto del alumno como del tutor del proyecto, además de las licencias de programas informáticos requeridos:

Tabla de la estimación del presupuesto del proyecto [obtención propia]:

Activo/producto utilizado, coste personal:	Coste:
Licencia Abaqus	6000 €
Microsoft office	130 €
Tutor	2800 € (80horas, 35€/hora)
Alumno	5250 € (350 horas aprox, 15€/hora)

Cabe destacar que como Python no es un programa en sí, sino un lenguaje de programación, su uso no ha significado coste alguno. El coste total estimado es de: 14180 €

Bibliografía:

[1] E. Giner Maravilla, J.E. Tarancón Caro, C. Arango Villegas, A.Vercher Martínez , F.J. Fuenmayor Fernández. *Estimación de propiedades elásticas y resistentes de la línea cementante en tejido óseo cortical a partir de ensayos experimentales y modelos de elementos finitos*

[2] fisiopatología osea: https://www.ucm.es/data/cont/docs/420-2014-02-18-01%20fisiopatologia%20osea.pdf

[3] Federico C. Buroni, Pablo E. Commisso, Adrián P. Cisilino y Mario Sammartino. *Determinación de las constantes elásticas anisótropas del tejido óseo utilizando tomografías computadas. Aplicación a la construcción de modelos de elementos finitos.*

[4] Élisa Budyn, Thierry Hoc. *Multiple scale modeling for cortical bone in fracture in tension using X-FEM*

[5] Simin Li, Adel Abdel-Wahab, Emrah Demirci, Vadim V. Silberschmidt.*Fracture process in cortical bone: X-FEM analysis of microstructured models*

[6] Tony M. Keaveny, Elise F. Morgan, Oscar C. Yeh. *Bone Mechanics*

[7] Joao Folgado, Paulo R. Fernandes. *Bone Tissue Mechanics*

[8] BME 332: Introduction to Biosolid Mechanics: http://www.umich.edu/~bme332/ch9bone/bme332bone.htm

[9] Hueso cortical y Hueso trabecular: http://inforosteoporosis.blogspot.com.es/2013/03/hueso-cortical-y-hueso-trabecular.html

[10] Hueso cortical (compacto): http://lasaludfamiliar.com/contenido/articolos-salud-262.htm

[11] Tejidos animales: conectivo, óseo. Compacto:

http://mmegias.webs.uvigo.es/a-imagenes-grandes/oseo_compacto.php

[12] Sistema Óseo: Estructura y función: http://www.iqb.es/cbasicas/fisio/cap06/cap6_1.htm#fisiologia

[13] Dr. José Sánchez. *Homeostasis y mineral ósea*: http://es.slideshare.net/johannafarez1/homeostasis-y-mineral-sea

[14]Ramón Contreras. *Los osteoclastos*: http://biologia.laguia2000.com/citologia/los-osteoclastos

[15] Pablo Morales. *El sistema de Havers*: http://biologia.laguia2000.com/citologia/el-sistema-de-havers

[16] Biología Ósea: http://bibing.us.es/proyectos/abreproy/70329/fichero/2+-+Biolog%EDa+%F3sea.pdf

[17] Mechanical properties of bone: http://www.doitpoms.ac.uk/tlplib/bones/bone_mechanical.php

[18] Instrumented Indentation:

http://www.npl.co.uk/science-technology/mass-and-force/hardness/instrumented-indentation

[19] Biomecánica: http://www.mibienestar.es/salud/2-general/2-biomecanica.html

[20] Diseases and conditions. Osteoporosis:

http://www.mayoclinic.org/diseases-conditions/osteoporosis/basics/definition/con-20019924

[21] Abaqus Simulia: http://www.3dcadportal.com/abaqus-simulia.html

[22] Abaqus unified FEA: http://www.3ds.com/products-services/simulia/products/abaqus/

[23] Eduardo Frias Valero. *El metodo de los elementos finitos*

[24] Maylett Y. Uzcátegui Flores. *Abaqus programa de elementos finitos*

[25] J.T.B. Overvelde. *Learn Abaqus script in one hour*

[26] Abaqus 6.12. Scripting User's manual

[27] Gautam M. puri. *Python scripts for Abaqus*

[28] Introduction to the Abaqus scripting interface: http://www.3ds.com/products-services/simulia/services/training-courses/course-descriptions/introduction-to-the-abaqus-scripting-interface/

[29] Python scripting tutorial: http://www.dreamsyssoft.com/python-scripting-tutorial/

[30] UNSW embryology: https://embryology.med.unsw.edu.au/embryology/index.php?title=File:Osteoclast.jpg

[31] más que ciencía. Matriz extracelular:

http://www.mas-que-ciencia.com/matriz-extracelular/

[32] Fisiopatología ósea: https://www.ucm.es/data/cont/docs/420-2014-03-28-01%20Fisiopatologia%20osea.pdf

[33] The skeleton, bones and joints: http://www.bbc.co.uk/schools/gcsebitesize/pe/appliedanatomy/2_anatomy_skeleton_rev4.shtml

[34] Ruperto Bastidas. Osteoporosis: http://slideplayer.es/slide/1091312/

[35] Que es el sistema de Havers: https://curiosoando.com/que-es-el-sistema-de-havers

[36] Empleo de la modelación por elemento finito (MEF) en la solución de problemas ingenieriles:

http://www.monografias.com/trabajos82/aplicacion-mef-soldadura/aplicacion-mef-soldadura2.shtml